MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A123 959	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* PERFORMANCE MODELING OF MULTIPROCESSOR SYSTEMS WITH PAGING		5. TYPE OF REPORT & PERIOD COVERED Technical Panel
		6. PERFORMING ORG. REPORT NUMBER R-892; UILU-ENG 80-2224
7. AUTHOR(s) Alan Dale Gant		8. CONTRACT OR GRANT NUMBER(s) N00014-79-C-0424
9. PERFORMING ORGANIZATION NAME AND ADDRESS Coordinated Science Laboratory University of Illinois at Urbana-Champaign Urbana, Illinois 61801		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Joint Services Electronics Program		12. REPORT DATE October, 1980
		13. NUMBER OF PAGES 91
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

multiprocessors, memory hierarchy,
paging system, shared secondary memory,
analytic modeling

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The rapid advancement in semiconductor technology continues to change the environment in which computers are designed. As hardware costs decline, systems with multiple processors become an interesting alternative to conventional single processor systems. An analytic model has been developed to describe the performance of a wide range of multiprocessor system configurations and workloads. This model deals specifically with P tightly-coupled, identical processors with shared primary and secondary memory. Secondary memory consists of a paging drum with S sectors. The workload consists of J

DD FORM 1473
1 JAN 73

20. ABSTRACT (continued)

lambda

independent, identically-distributed jobs whose faulting or I/O behavior is described by both a mean (λ faults/sector-time) and a squared coefficient of variation(K). In addition, the processing overhead for each I/O request is added to a job's execution time at a processor(C sector-times/fault).

The model developed provides an estimate of system throughput for various numbers of processors, jobs, and drum sectors, and for various workloads. Throughput, the average number of processors doing useful work, is given by

$$T = \min\left\{ \frac{\frac{1}{\lambda}+C+\frac{S}{2}+1+J -\sqrt{\left(\frac{1}{\lambda}+C+\frac{S}{2}+1+J\right)^2 -4J\left(\frac{1}{\lambda}+C+1\right)}}{2\lambda\left(\frac{1}{\lambda}+C+1\right)} , \frac{P\frac{1}{\lambda}}{\frac{1}{\lambda}+C} \right\} .$$

This model is based on a deterministic scheduling model for the system, and known models which describe the sub-parts of the system are embedded. The accuracy of the model is assessed by comparison to a large number of runs of a simulator using exponential and hyperexponential fault time distributions. For a wide range of values of the parameters, the formula provides a very good estimate of throughput (the average relative error for 195 simulation runs is only 3.0%.). Even though the squared coefficient of the fault distribution varied from 1 to 16 in the simulation runs, the model fit quite well without using K. This suggests that the mean fault rate is perhaps a sufficient measure of the faulting process. Further evidence for this conclusion is the fact that the model's prediction only improves slightly when a drum queue wait model is employed which includes K.

The model can be used to examine the behavior of multiprocessor systems, including the sensitivity of system throughput to each of the system parameters and parameter trade-offs related to system performance. In particular, in a system with a fixed amount of memory, the addition of jobs to the system causes a change in the memory allocation for each job and thus modifies each job's faulting behavior. The above formula for throughput is useful to examine the desirability of adding or subtracting jobs in such a system.

PERFORMANCE MODELING OF MULTIPROCESSOR SYSTEMS WITH PAGING

by

Alan Dale Gant

PERFORMANCE MODELING OF MULTIPROCESSOR SYSTEMS WITH PAGING

BY

ALAN DALE GANT

B.S., University of Texas, 1974
M.S., University of Illinois, 1977

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1980

Urbana, Illinois

PERFORMANCE MODELING OF MULTIPROCESSOR SYSTEMS WITH PAGING

Alan Dale Gant, Ph.D.
Department of Electrical Engineering
University of Illinois at Urbana-Champaign, 1980

The rapid advancement in semiconductor technology continues to change the environment in which computers are designed. As hardware costs decline, systems with multiple processors become an interesting alternative to conventional single processor systems. An analytic model has been developed to describe the performance of a wide range of multiprocessor system configurations and workloads. This model deals specifically with P tightly-coupled, identical processors with shared primary and secondary memory. Secondary memory consists of a paging drum with S sectors. The workload consists of J independent, identically-distributed jobs whose faulting or I/O behavior is described by both a mean (λ faults/sector-time) and a squared coefficient of variation(K). In addition, the processing overhead for each I/O request is added to a job's execution time at a processor(C sector-times/fault).

The model developed provides an estimate of system throughput for various numbers of processors, jobs, and drum sectors, and for various workloads. Throughput, the average number of processors doing useful work, is given by

$$T = \min\left\{ \frac{\frac{1}{\lambda}+C+\frac{S}{2}+1+J - \sqrt{(\frac{1}{\lambda}+C+\frac{S}{2}+1+J)^2 - 4J(\frac{1}{\lambda}+C+1)}}{2\lambda(\frac{1}{\lambda}+C+1)} \; , \; \frac{P\frac{1}{\lambda}}{\frac{1}{\lambda}+C} \right\} \; .$$

v

This model is based on a deterministic scheduling model for the system, and known models which describe the sub-parts of the system are embedded. The accuracy of the model is assessed by comparison to a large number of runs of a simulator using exponential and hyperexponential fault time distributions. For a wide range of values of the parameters, the formula provides a very good estimate of throughput (the average relative error for 195 simulation runs is only 3.0%.). Even though the squared coefficient of the fault distribution varied from 1 to 16 in the simulation runs, the model fit quite well without using K. This suggests that the mean fault rate is perhaps a sufficient measure of the faulting process. Further evidence for this conclusion is the fact that the model's prediction only improves slightly when a drum queue wait model is employed which includes K.

The model can be used to examine the behavior of multiprocessor systems, including the sensitivity of system throughput to each of the system parameters and parameter trade-offs related to system performance. In particular, in a system with a fixed amount of memory, the addition of jobs to the system causes a change in the memory allocation for each job and thus modifies each job's faulting behavior. The above formula for throughput is useful to examine the desirability of adding or subtracting jobs in such a system.

ACKNOWLEDGMENT

TABLE OF CONTENTS

1. INTRODUCTION AND BACKGROUND

1.1 Problem Description

As technology continues to alter design constraints and reduce hardware costs of computers, systems with multiple processors become an increasingly interesting alternative to conventional single processor systems. In order to evaluate the performance of multiprocessor systems, an analytic model has been developed which describes the system performance as a function of page faulting behavior. In particular, the model deals with systems of P processors with shared memory, a paging drum (or rotating or electronic equivalent) with S sectors, and a work load consisting of J independent jobs. The effects of various design decisions and trade-offs on system throughput have been examined directly using the analytic model. The accuracy of the model has been evaluated with a large number of simulations of the system.

1.2 System Configuration

In order to analyze the performance of multiprocessor computer systems with paged memory, a specific system configuration was studied (Figure 1-1). The specification of this paged memory system includes
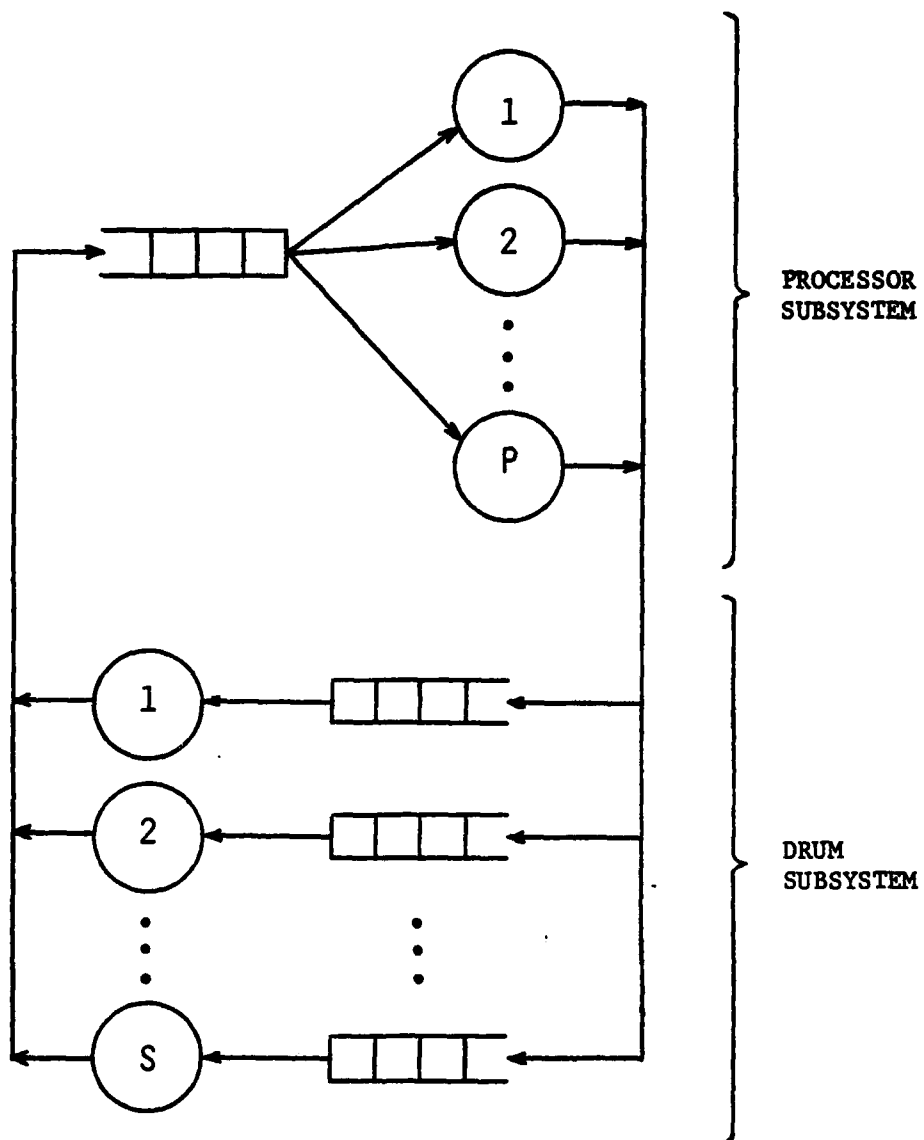
Figure 1-1. System Configuration.

both virtual memory systems with demand paging and systems using explicit paged I/O.

In particular, the processor subsystem consists of P identical, independent processors. These processors share a single memory in order to share one job load. The specifics of the processor-memory interconnection is not known except that it is assumed that access conflicts among processors can be neglected, i.e., that their effect on performance can be accounted for by a simple adjustment of a model parameter value. This assumption is possible since the model is concerned with the performance of the system at the level of page faults, which hopefully occur at intervals of large numbers of memory accesses. Also, memory organizations have been identified which achieve very low incidences of conflict [BRIG77a, BRIG77b].

As jobs fault, they leave the processor subsystem and travel to secondary memory to fetch the needed pages. The secondary memory is assumed to be a sectored drum. This drum may actually be constructed as a head per track disk or some semiconductor analogue such as bubble or charge-coupled device (CCD) memory. Drum is intended here to represent any secondary memory whose behavior may be modeled as a queueing delay and a rotational latency followed by the transfer of data. Moving head disks are not modeled since they also require a seek time.

In addition, each sector of the drum maintains a queue of jobs awaiting service. Sector queues allow a shortest latency time first (or SLTF) schedule to be used. Jobs requesting service at a specific sector are serviced in the order of their arrival, but jobs at the heads of different sector queues are serviced in the order that the sectors can be read.

In a virtual memory demand paging system, some of the page faults may require a page of memory to be written to the drum before the needed page can be read into memory. The model used here does not specifically deal with such writes. However, the model is still valid under the following conditions. It is possible for the drum controller to buffer these drum writes separately from the reads. Then the transfer of needed pages to memory can be given priority over writes. This operation allows the jobs which have faulted to return to the processor subsystem as quickly as possible. The lower priority drum writes can be accomplished whenever a particular drum sector has no pending reads. Of course, the drum controller must manage the additional bookkeeping of checking to see if any drum read requests refer to a drum write which has not been completed. An alternate approach would be to prewrite some pages which have been changed when the drum is idle. Then there would be no need to write these pages to the drum when they are replaced by new pages. If the system model is used to describe general I/O requests to a paging drum instead of a virtual memory system, then the occurrence of a model "page fault" represents a single page read or write. In this

case, the controller can treat reads and writes identically. To accurately model a virtual memory demand paging system without a sophisticated controller capable of prioritizing reads and writes, an additional parameter would have to be added to describe the occurrences of writes before reads, and the model would need to place both drum service requests in drum queues.

As jobs return to the processor subsystem from the drum they enter a single first in first out, FIFO, queue to await a free processor. The number of jobs in the system is assumed to be constant, as shown for P processors with an S sector drum in the diagram of Figure 1-1. When a job finally completes execution and leaves the system it is immediately replaced by a similar job which is assumed to be waiting outside the system. Thus the system under study is a closed queueing network.

The system may be described by means of several parameters which specify the number of elements in the system and their behavior. All references to time in the system will be in units of a <u>sector-time</u>, where one sector-time is the time taken for one sector of the drum to pass by the drum heads. The parameters used for our model are:

S   the number of drum sectors (pages/track),

P   the number of processors in the system,

J   the number of jobs in the system,

M   the number of words of memory,

W   the number of words per page,

C   the operating system overhead time per page fault,

λ the mean of the fault rate distribution

(faults/sector-time), and

K   the squared coefficient of variation of the

fault rate distribution.

1.3 Analysis Techniques

Numerous methods are available to study the performance of a computer system. In this section, some of these are discussed and their applicability to the problem under study is considered.

An obvious method for determining system performance is the use of simulation, which allows a precise description of the operation of the system components and their interaction. However, the results of a simulation for a specific configuration of the system and a specific workload are valid only for the specific case simulated. Thus a single run of the simulation describes the performance of the system for one value of each of the system parameters. To perceive overall trends in performance, each of the system parameters must take on several values independently of the other parameters. This fact suggests, for example, that i parameters each taking on j values would require as many as j^i runs of the simulation. Also, since each of the simulation runs yields only an isolated numerical result, the underlying mechanisms causing variations in system performance are difficult to determine. Nevertheless, a collection of simulations may be extremely useful for

checking the accuracy of analytic models.

To reduce the cost of simulation of the system, certain simplifications of system operation might be made. For instance, the detailed operation of a subsystem of the system might be replaced by an approximate model of the subsystem. Of course, the accuracy of the simulation results would depend on the accuracy of the approximate model.

Another approach for modeling such a system is a network of queues and servers, as was shown in Figure 1-1 [JACK63,CHAN77]. The solution of queueing networks is possible if the servers all have exponential service time distributions. Solutions are also available for non-exponential service times if an immediate service discipline is used, e.g., last come first served (LCFS). Since neither SLTF nor FIFO fall into the immediate service disciplines and since the drum does not have exponential service, these models are not directly applicable to the system under study.

Some recent work in queueing theory has been directed toward using various approximation techniques in order to model systems which do not fall into the category of "well-behaved" systems which can be explicitly described by conventional queueing theory methods. One method which has recently been developed is the application of Norton's theorem [CHAN75]. For closed networks consisting only of exponential servers, an analog of Norton's theorem from electrical circuit theory can be used to replace a

subset of the queues with a single queue. Thus a network can be reduced in complexity. This technique has been extended to provide an approximate analysis in cases where servers are non-exponential.

Another approximation technique recently proposed is the division of distributions into percentiles [LAZO77]. Reducing the precision of the description of the distributions is a brute force technique designed to achieve some results under difficult conditions.

In addition, much recent work has been based on the diffusion approximation [KOBA74, GELE75]. Here, the discrete flow of jobs through the queues is replaced by a continuous "diffusion" model. This model most accurately describes systems under a heavy traffic assumption where servers are very seldom idle.

For all of the approximation techniques of queueing theory there is no easily computed measure of their accuracy. In order to use them, their accuracy must be verified by other means, such as simulation of the system.

Another general method for modeling the system is to devise analytic models based on appropriate simplifications of system operation. Such a heuristic-driven approach, like queueing theory approximations, normally has unknown accuracy, but can be checked by simulation. Nevertheless, approximate analytic models may yield equations which detail the operation and interaction of system

components in a concise and informative manner.

In this research, a combination of two of the above methods is used. Detailed simulations of the system under various workloads and system configurations are made to provide a precise knowledge of system operation for these cases. The details of these simulations and the assumptions made are discussed in section 2.1. Then, analytic models are derived and adapted from the literature. Since the goal of this research is to accurately model the paging behavior of the system, any technique or combination of techniques is potentially useful. In particular, queueing theory models, both exact and approximate, are used for certain subsystems. The system model is developed from a deterministic scheduling discipline. The accuracy of the models is checked with the results of the simulations. In this way, a precise description of system operation is provided by the simulations and the analytic models provide a concise means of discerning relationships between performance and the system parameters and underlying causes of performance variations.

1.4 Processor Models

The processor subsystem consists of the P processors and their shared input queue(Figure 1-1). Several models already exist which may be useful for describing a multiple processor subsystem. If an infinite number of jobs is available to the processors, the departure of jobs to

the drum after page faults occur is solely dependent on the service times of the processors. In particular, if exponential service times are assumed, the flow of jobs leaving P independent processors is the confluence of the job flows leaving each processor, each of which constitutes a Poisson process. If the mean fault rate of all the jobs is the same, λ , then the flow of jobs to the drum is a Poisson process with mean rate P λ. This rather simple description may be useful in determining the "bottleneck" of a system. The service rate of the drum is one page, or job, per sector-time. Thus, the service rate of the drum subsystem, 1 job/sector-time, may be compared to the mean service rate of the processor subsystem, P λ, to estimate behavior under heavy load.

Another simple model of the processor subsystem involves the use of an M/M/1 queueing model. Here, all P processors are replaced by a single processor with equivalent throughput. This effect is achieved by assuming a processor sharing discipline, in which all running jobs share the P processors equally (often described as timesharing with an infinitesimal time slice) [COFF73]. The single processor has an exponential fault distribution. This model also assumes a Poisson arrival process. In fact, the arrival process to the processor subsystem is unknown and unlikely to be Poisson.

The processor subsystem might also be modeled by an M/M/P queue, which extends the M/M/1 model to P parallel servers. However, this model is extremely complex and even simple measures of performance are not easily found [COFF73, KLEI75]. This approach is also restricted to an exponential fault distribution and a Poisson arrival process.

1.5 Drum Models

The drum subsystem consists of the drum and the S sector queues as in Figure 1-1. A drum, with fixed-length sectors, operating with an SLTF (shortest latency time first) schedule has been examined previously, and two models are of interest here.

First, an expression for the the expected waiting time, Q_d, for a paging drum has been derived by Coffman [COFF69] and by Fuller and Baskett [FULL75], under the restriction of Poisson arrivals with mean rate λ_d. The utilization of the drum, ρ_d, is the arrival rate divided by the service rate. Since the service rate of the drum is 1 job/sector-time, the utilization is equal to the arrival rate, λ_d. The expression for the expected time spent at the drum (including queue wait but neglecting service time) is

$$Q_d = \frac{S}{2} + \frac{S\lambda_d}{2(1 - \lambda_d)} \quad . \tag{1-1}$$

The first term in equation 1-1 denotes an average one-half revolution

that a job waits for its sector to be reached, and the second term describes the average time spent in the sector queue while waiting for previously arrived jobs to be serviced. The S coefficient is the rotation time of the drum, and as λ_d approaches the service rate of the drum, 1, the magnitude of the second term approaches infinity.

Obviously, arrivals to the drum subsystem may not be Poisson. Adams, Gelenbe, and Vicard have developed a model for the SLTF paging drum which utilizes the mean of the arrival rate to the drum as well as its squared coefficient of variation [ADAM79]. In order to deal effectively with non-Poisson arrivals, they use the diffusion approximation developed by Gelenbe [GELE75]. Since the diffusion approximation has unknown accuracy, they compare the results of their model with data supplied by simulations using two-stage hyperexponential distributions to provide drum arrivals with variance greater than or equal to that of an exponential distribution. A two-stage hyperexponential is sufficient to generate distributions with any mean and any variance which is greater than the mean. For high values of drum utilization, the model agrees reasonably well with their simulations. The equation for mean time spent in the drum queue with mean arrival rate λ_d and squared coefficient of variation K_d is

$$Q_d = \frac{(S + K_d - 1)\lambda_d}{2(1 - \lambda_d)} \ . \tag{1-2}$$

This formula is similar to that derived by Coffman. It differs in two ways. First, it includes a correction for K_d not equal to 1 (the exponential distribution has a K of 1) and second, it does not include an S/2 term for the one-half revolution average wait for the sector in question to reach the read head. The heavy traffic assumption made in applying the diffusion approximation implies that the value of equation 1-2 would predominate over S/2, since λ_d approaches 1 as ρ_d approaches 1. The coefficient, $S + K_d - 1$, denotes an "effective" rotation time due to the change in variance of the arrivals to the drum.

1.6 Program Behavior

Along with the processor subsystem and the drum subsystem, the job workload determines system performance. The description of how jobs behave on a computer system is called program behavior. Of particular interest here is that part of program behavior related to paging.

The paging behavior of a program is dependent on the internal structure of the program and on the configuration of the machine on which that program runs. In particular, the page size, W, the size of memory, M, and the number of jobs, J, combine to dictate the average allocation of pages that a job receives. The paging policy of the system also determines whether that allocation is fixed in size or varies dynamically, such as when using the working set algorithm [DENN68] or the page fault frequency algorithm [CHU72]. Studies of

dynamic allocation schemes have shown them to be of widely varying efficiency [BUDZ77].

Numerous studies have been made of the paging behavior of programs. The effects of page size and memory allocation on page fault behavior have been studied by Kuck and Lawrie [KUCK70]. They also examined the effects of the number of jobs in the system for a single processor system. Terms such as locality and working set have been used to describe the tendency of programs to reference a particular subset of their address space over a short section of their execution. For instance, Ferrari has developed the notion of bounded locality intervals, which are in effect nested localities of decreasing size [FERR76]. His work suggests that there are in actuality several localities of varying size in existence at any one time. To date, however, there appears to be no concise quantitative description of program paging behavior as a function of relevant, measurable system and program attributes which is representative of programs in general.

1.7 Model Decomposition

The system described in Figure 1-1, along with the parameters defined in section 1.2, details both the physical organization of the computer system and the paging characteristics of the job workload. The system under study includes P processors, J jobs, S drum sectors, M words of memory, and W words per page. C denotes the overhead time

required to service a page fault. This unified model of the performance of the system is symbolized in Figure 1-2a. In order to reduce the complexity of the model, it may be decomposed into two sub-models connected in a serial fashion (Figure 1-2b). Here, the characteristics of a job's faulting behavior are dealt with in a fault model, which considers the memory configuration, the number of jobs sharing the memory space, and the paging behavior of the workload. The output of this first sub-model, a descriptor of program faulting behavior, is then used as input to the system model. Besides this fault descriptor, parameters describing the physical configuration of the system, the number of jobs in the system, and the operating system overhead time per fault, C, are input to the system model. The performance of the system is obtained from the system model.

As mentioned in section 1.6, the faulting behavior of programs in general is not well understood. In fact, no model such as the fault model of Figure 1-2b has been developed. However, development of the second sub-model may proceed somewhat independent of the first. If at a later time an accurate fault model which is different from λ and K is developed, the system model may require adjustment to accept this new fault descriptor. In the meantime, results from a system model alone can be used to develop intuition about system performance as a function of J, P, S, C, and the fault descriptor as well as to help identify the essential characteristics of a useful fault model.
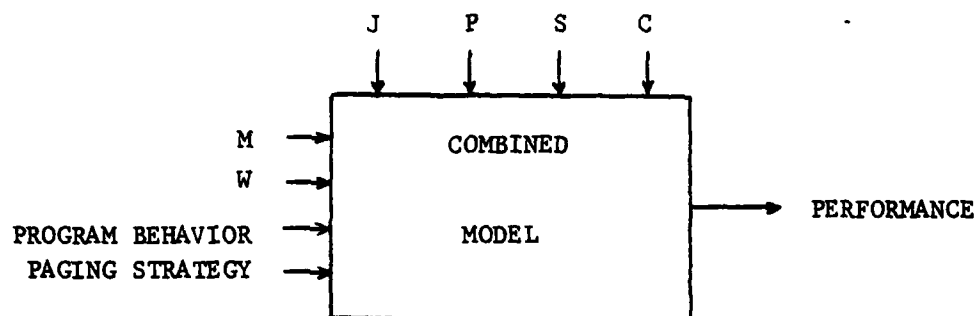
(a)
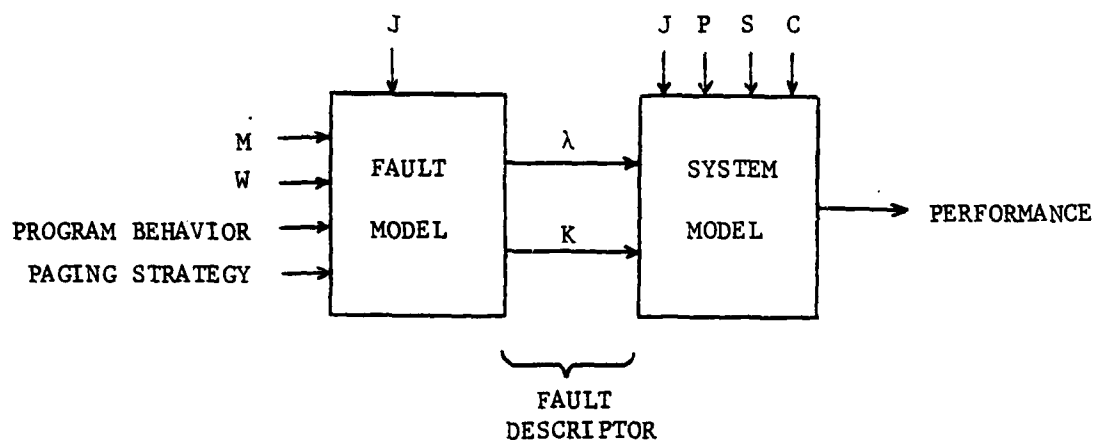
(b)

Figure 1-2. Model Decomposition.

The remainder of this research will deal only with the system model. The faulting of the job workload will be assumed to be representable by various probability distributions, and the fault descriptor will be a function of the first two moments of these distributions. In particular, the mean, λ , and the squared coefficient of variation, K, will together describe the faulting process. By including a measure of the variance of the process in addition to the mean, a wider range of faulting behavior can be studied.

2. MODEL DEVELOPMENT

The goal of this chapter is to develop an effective model of system performance. Performance is measured in terms of throughput, T, which is defined to be the average number of processors in the system doing useful work. Any time a processor spends in operating system overhead does not constitute useful work. The approach used involves the use of existing and new subsystem models, which are then utilized in a single model for system performance. This system model is evaluated by comparison to performance data supplied by detailed simulations of the system.

Analytic models are especially attractive because they provide insight into the functional dependencies of the system. In particular, it is possible to discern directly the effects on throughput, T, of any system parameters, singly or in combination. With this knowledge comes a better understanding of the underlying mechanisms of system operation, rather than just the numerical tables provided by simulation alone.

The analytic models derived in this chapter require certain assumptions about and simplifications of the system. As such, their accuracy is questionable. In order to check accuracy, a simulator was written, in SIMULA, to obtain precise values of system performance for a

range of values of the system parameters [BIRT73, CDC75].

Chapter 2 first describes the details of the simulation of the system and then proceeds to develop the analytic models. In particular, a model based on deterministic scheduling is derived and augmented with models of the paging drum subsystem. Finally, models for processor queue delay are discussed.

2.1 System Simulation

In order to check the accuracy of analytic models of the system and provide insight into the operation of multiple processor systems, a simulator was written to provide a large performance data base. A summary of the data from each simulation run is provided in the appendix. Written in SIMULA, the simulator accurately describes the flow of jobs through the system. Jobs are represented by a fault distribution. The simulator draws fault times from a given interfault time probability distribution by using a random number generator. The actual operation of the drum is described. Each sector queue is dealt with as it rotates by the drum heads. Faulted jobs leaving the processors are assumed to be uniformly distributed among the drum sector queues. Thus the simulator uses another random number generator to assign faulted jobs to sector queues. Inputs to the simulator include the system parameters S, P, J, and C; fault distribution parameters λ, and K; and seeds for the various random number generators. The

simulator collects and prints statistics related to the operation of the system. Measures collected for the processors include processor busy time, idle time, and overhead time. Busy time refers only to the time a processor is busy doing useful work. For each queue in the system, the mean length and the mean time a job spends in the queue are determined. For the drum, the utilization and total number of faults serviced are recorded. The flow of jobs from the processors to the drum subsystem is characterized by its mean and its squared coefficient of variation. In addition, statistics such as the observed fault rate are taken to check on the distributions derived from the random number generators.

Numerous configurations of the system were simulated. The number of processors, P, the number of jobs, J, the number of sectors, S, and the overhead time per fault were widely varied. The faulting behavior of the jobs was defined either by an exponential distribution or a two-stage hyperexponential distribution, so as to cover a wide range of behavior. The mean fault rate for the fault distributions was also greatly varied. In all, 195 different simulation runs were completed. An attempt was made to cover the range of system behavior from lightly loaded to saturated as each of the parameters was varied. Also, most simulation runs executed for several thousand faults. The primary exceptions to this runtime were some two-stage hyperexponential distributions. The two distributions differed by several orders of magnitude and the slower of the two only generated around 500 faults, but the total number of faults generated was over 20,000.

2.2 System Model

2.2.1 Deterministic Schedule Model

As a job flows through the system, it visits four sites in the
system: one of the P processors for useful runtime and for fault
overhead (the time spent by the operating system to service a fault),
one of the S sector queues, the drum for actual transport of the page
needed, and the processor queue to wait for a free processor. The time
that a job takes to travel once around the system may be called its trip
time. In general then, the average trip time, t, is given by

$$t = \frac{1}{\lambda} + C + 1 + Q_d + Q_p \, , \qquad (2-1)$$

where Q_d and Q_p are the mean wait times at the drum and processor
queues, respectively. If the restriction is made that the time spent at
each of these sites is constant and equal to the mean time spent at that
site in the idealized probabilistic system, a deterministic schedule may
be applied to determine the system throughput.

In such a deterministic system, there are two regions of operation,
processor saturated and processor unsaturated. When the processors are
saturated, or continuously occupied, each processor provides a job $1/\lambda$
useful runtime followed by C overhead time before starting the next job.
Thus, the throughput is equal to the percentage of time each processor

spends doing useful work, $(1/\lambda)/((1/\lambda) + C)$, times the number of processors, P. Since the maximum throughput of the system occurs when the processors are saturated, then

$$T \leq \frac{P}{\lambda(\frac{1}{\lambda} + C)} \quad . \tag{2-2}$$

If the processors are unsaturated, then no job returning to the processor subsystem need wait for a processor to become free. In this case, Q_p, the processor queue wait, will be zero. The trip time, t, is now given by

$$t = \frac{1}{\lambda} + C + 1 + Q_d \quad . \tag{2-3}$$

In one trip time, a single job receives one interfault time of useful service, $1/\lambda$. For J jobs in the system with no interference, J/λ useful work is done in each trip time, t, so the throughput of the system is upper bounded by $J/(\lambda t)$.

Since each case gives an upper bound, the actual throughput, T, is then upper bounded by the minimum of the two cases:

$$T \leq \frac{\frac{PJ}{\lambda}}{\max[J(\frac{1}{\lambda} + C), \ P(\frac{1}{\lambda} + C + 1 + Q_d)]} \quad . \tag{2-4}$$

To show that the throughput for deterministic scheduling with constant

service and queueing times is exactly equal to equation 2-4, it is only necessary to show the existence of one schedule which will achieve that throughput.

A schedule which achieves this maximum throughput is illustrated in Figure 2-1. First, run all jobs, 1 to J, in succession on P_1, the first processor. As each job completes its $(1/\lambda) + C$ runtime, place it at the drum for a time of $Q_d + 1$. Then place each job in the processor queue for Q_p. Note that for a deterministic system, Q_p is zero if the processors are unsaturated. As jobs return to the processor subsystem, run them in succession on processor P_2, followed by the same wait at the drum and at the processor queue. Continue this procedure on successive processors through P_p. Then restart the schedule on P_1.

Now, consider the time, Pt, between successive restarts of job 1 on processor P_1. This time is $P((1/\lambda) + C + 1 + Q_d)$ in the case of unsaturated processors (Figure 2-1a) and is $J((1/\lambda) + C)$ in the processor saturated case (Figure 2-1b). The total amount of useful runtime during the cycle is PJ/λ in either case. This analysis results in the throughput specified in equation 2-4, so the throughput of the system is given by

$$T = \frac{\frac{PJ}{\lambda}}{\max[J(\frac{1}{\lambda}+C),\ P(\frac{1}{\lambda} + C + 1 + Q_d)]}\ . \tag{2-5}$$

The condition for saturation of the processors is given by

P_1 1 2 3 $\cdots$ J idle 1 2 3 $\cdots$ J idle 1 2 3 $\cdots$

P_2 1 2 3 $\cdots$ J idle 1 2 3 $\cdots$ J idle 1 2 3 $\cdots$

P_3 1 2 3 $\cdots$ J idle 1 2 3 $\cdots$ J idle $\cdots$

P_p 1 2 3 $\cdots$ J idle 1 2 3 $\cdots$ J $\cdots$

$\frac{1}{\lambda} + C$ $1 + Q_d$ Pt
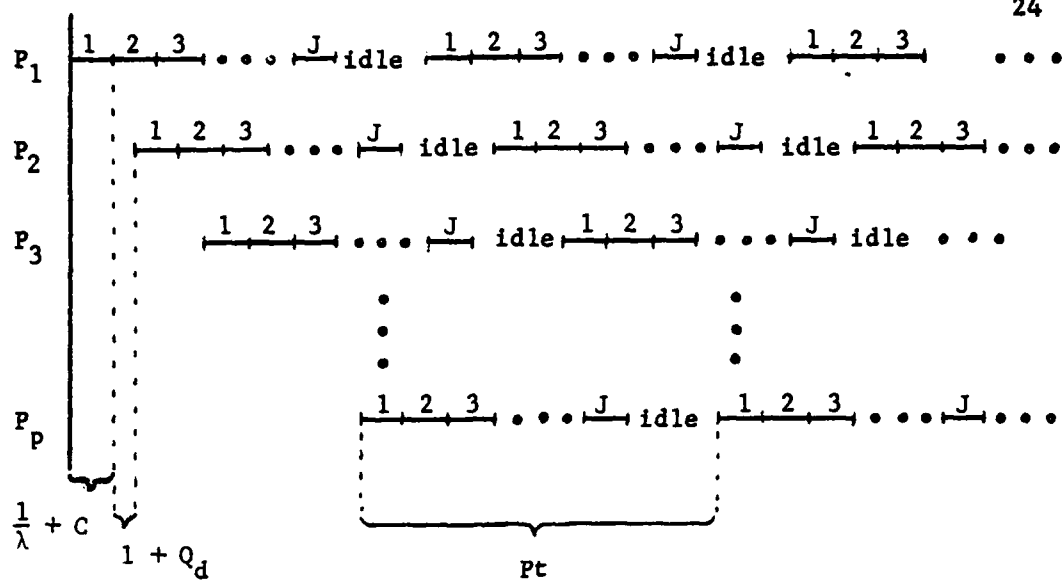
(a)

Processors Unsaturated

P_1 1 2 3 $\cdots$ J 1 2 3 $\cdots$

P_2 1 2 3 $\cdots$ J 1 2 3 $\cdots$

P_3 1 2 3 $\cdots$ J 1 2 3 $\cdots$

P_p 1 2 3 $\cdots$ J 1 $\cdots$

$\frac{1}{\lambda} + C$ $1 + Q_d + Q_p$ Pt

(b)

Processors Saturated

Figure 2-1. Maximum Throughput Schedule.

$$J \geq \frac{P(\frac{1}{\lambda} + C + 1 + Q_d)}{\frac{1}{\lambda} + C} \quad . \tag{2-6}$$

The numerator of the right hand side represents the total available processor execution time over one trip time. J multiplied by the denominator of the right hand side is the total processor execution time needed by the J jobs in one trip time. Thus, if the total time needed by the J jobs exceeds the total time available from the P processors, the processors will be saturated.

2.2.2 Application of Deterministic Model

The deterministic model described above is exact for a system in which all events occur at constant time intervals. But in the system under study, the operation of the system components is determined probabilistically. In particular, the service time of a job at a processor varies probabilistically. This variation in turn affects the flow of jobs around the system and thus also affects the time that jobs spend waiting in the queues. In particular, it is possible to have non-zero processor queue wait, Q_p, even though the processors are unsaturated. This phenomenon occurs when the processors are saturated part of the time but not always. When Q_p is included in a job's trip time, the model throughput becomes

$$T = \frac{\frac{PJ}{\lambda}}{\max[J(\frac{1}{\lambda} + C), \ P(\frac{1}{\lambda} + C + 1 + Q_d + Q_p)]} \quad . \tag{2-7}$$
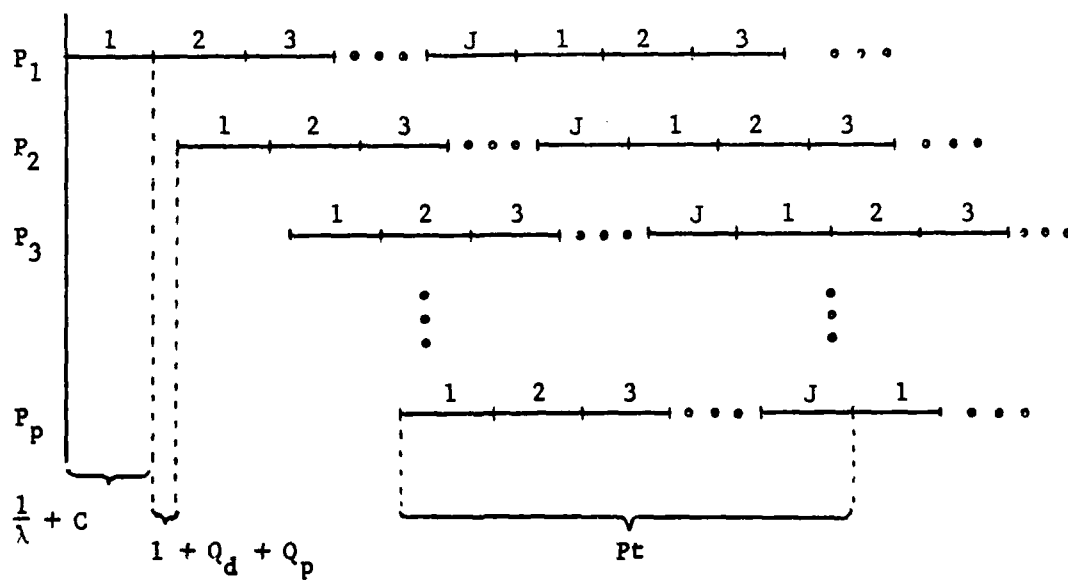
This version of the model will be referred to as the deterministic model from here on. The deterministic model is exact for the probabilistic system if the λ, C, Q_d, and Q_p values used are exact averages. Accurate estimates of Q_d and Q_p need to be expressed as functions of the basic parameters, while λ and C are assumed as basic parameters. In fact, if Q_p and Q_d are accurate, the unsaturated case will always accurately predict throughput, because Q_p will increase as the system moves into saturation and cause both terms in the denominator of equation 2-7 to be equal. The limit imposed by the saturated case is only necessary to avoid overestimation of throughput by an inaccurate model for Q_p in the unsaturated case.

To evaluate the accuracy of the deterministic model, values were selected for the parameters of the model and the throughput predicted by the model was compared to the throughput determined by the simulations of the system. For the service times, the mean time spent at the server is used. Thus, the service time at a processor is taken to be the sum of the mean interfault time and the operating system overhead time, $(1/\lambda)$ + C. For the drum, the service time is simply 1 job/sector-time. The values for Q_p and Q_d are taken from the data collected by the simulations. In this way, the closeness of the model's prediction of throughput to the simulation results may be ascertained, under the assumption that both Q_p and Q_d can be accurately modeled later. All errors will be expressed as relative errors, i.e., the difference between the model value and the simulation value, divided by the

simulation value.

When a comparison is made between the throughput obtained from the model using Q_d and Q_p from the simulation and the throughput provided by the simulator, a very good fit is obtained. In particular, the maximum percentage error in throughput over the 195 points of data from simulation runs is 6.5%, and the average error is only 0.98%. Ninety percent of the cases have an error less than 3.0% (This error will be referred to as the 90% point error.). This error results primarily from the accuracy of some of the simulation runs. In particular, when large interfault times are simulated the cost of each run increases drastically, and the total number of faults simulated is not as large as for other cases. For large interfault time cases, the measured interfault time from the simulation sometimes differs noticeably from the input value. To check this effect the actual interfault time which occurred for each simulation run can be used in the model. Because of the end effect of the simulation, an exact value for interfault cannot be obtained, but minimum and maximum values can. When each of these is used in the model, the throughput reported by the simulator falls between the minimum and maximum throughput predicted in 169 of the 195 cases. For an additional 19 cases the error is only plus or minus 0.01, which is the least significant digit reported by the simulation. The predicted throughput is off by more than 0.01 in only 7 cases, and the highest error is 0.09 out of 6.99, or a relative error of 1.3%. This suggests that the form of the deterministic model is quite accurate at

predicting system throughput. If we accept such an error level, it now remains to find suitable formulas to predict both Q_p and Q_d.

2.3 Drum Queue Wait Models

For the paging drum with a queue per sector, two models were discussed in section 1.5. Coffman obtained an accurate model for the time spent waiting in the queue when the arrivals to the drum are a Poisson process. Adams, Gelenbe, and Vicard derived an approximate model extended for non-Poisson drum arrivals characterized by their mean and squared coefficient of variation. These models were both examined as candidates for Q_d in the deterministic system model.

Repeating equation 1-1, Coffman's model for drum queue wait, Q_d, is

$$Q_d = \frac{S}{2} + \frac{S\lambda_d}{2(1 - \lambda_d)} \quad . \tag{2-8}$$

Although the arrivals to the drum may not be Poisson distributed, Coffman's model may be a good estimate of Q_d. The only unknown in equation 2-8 is λ_d, the mean arrival rate to the drum. If the throughput, T, of the system were known, however, the mean flow of jobs from the processor subsystem to the drum could be calculated. Given that T processors on the average are busy doing useful work and that the mean flow rate from a usefully busy processor is λ, then the mean flow rate from the processor subsystem is

$$\lambda_d = T\lambda \quad . \tag{2-9}$$

Using this value for λ_d gives

$$Q_d = \frac{S}{2} + \frac{ST\lambda}{2(1 - T\lambda)} \quad . \tag{2-10}$$

The drum model developed by Adams, Gelenbe, and Vicard, denoted by AGV, is a source of potential improvement because it considers the first two moments of the drum arrival process. As mentioned in Chapter 1, the AGV model does not include the rotational latency term, S/2. Without this term, the model predicts drum queue wait well only when the wait is significantly larger than S/2. Since the form of the AGV model closely follows that of Coffman's model, an S/2 term can be added to improve the prediction of Q_d under lightly loaded conditions. Note that when $K_d = 1$, such as for Poisson arrivals, this augmented model reduces to Coffman's model. A general form for the drum queue wait of both models is

$$Q_d = \frac{S}{2} + \frac{(S + K_d - 1)T\lambda}{2(1 - T\lambda)} \quad , \tag{2-11}$$

with K_d set equal to 1 for Coffman's model. Q_d from equation 2-11 (using the value of throughput from the simulations) can be compared to the value of Q_d from the simulations. When K_d is set equal to 1 (Coffman's model) the average error over all of the simulation runs is

13.7%. Using K_d derived from the simulations (AGV model), this average error is reduced to 10.5%. The 90% point error is 33.2% for the Coffman model and 24.0% for the AGV model. When Q_d is small, the error is less than this average figure. The worst predictions occur for large values of Q_d, and the maximum error is 360% for Coffman's model and 322% for the AGV model. Note, however, that large values of Q_d occur when the denominator of equation 2-11 is small and there is great sensitivity to changes in T. The estimation of Q_d will be improved using the deterministic model for T, later in this section. Preliminary error figures are summarized in Table 2-1.

To examine the effects of these drum queue models upon the deterministic model, equation 2-11 may be substituted into the deterministic model (equation 2-7) for Q_d, yielding T as a function of T. This new formula results in a quadratic in T for the unsaturated case:

$$\lambda(\frac{1}{\lambda} + C + 1 + Q_p - \frac{K_d-1}{2})T^2 - (\frac{1}{\lambda} + C + \frac{S}{2} + 1 + Q_p + J)T + \frac{J}{\lambda} = 0 \quad . \quad (2\text{-}12)$$

The roots of this formula are

$$T = \frac{A + \frac{S}{2} + J \pm \sqrt{(A + \frac{S}{2} + J)^2 - 4J(A - \frac{K_d-1}{2})}}{2\lambda(A - \frac{K_d-1}{2})} \quad , \quad (2\text{-}13)$$

where $A = 1/\lambda + C + 1 + Q_p$. But only one of these roots is interesting. By examining the formula for Q_d, equation 2-10, it is

Table 2-1.

Preliminary Drum Queue Model Accuracy

(see equation 2-11)

Model	Source of T	Q_d Error (in %)		
		Maximum	Average	90% Point
Coffman	simulation	360	13.7	33.2
AGV	simulation	322	10.5	24.0

obvious that a value of T which is greater than $1/\lambda$ causes Q_d to be negative. In other words, it creates a situation in which $\lambda_d = T\lambda > 1$, or the arrival rate to the drum exceeds the service rate of the drum. Careful examination of equation 2-13 reveals that the plus sign root always yields a value of T greater than $1/\lambda$, which can be shown in the following manner. Suppose the plus-sign root of T is not greater than $1/\lambda$ and show a contradiction, i.e.,

$$\frac{A+\frac{S}{2}+J+\sqrt{(A+\frac{S}{2}+J)^2-4J(A-\frac{K_d-1}{2})}}{2\lambda(A-\frac{K_d-1}{2})} \leq \frac{1}{\lambda} , \tag{2-14}$$

where $A = 1/\lambda + C + 1 + Q_p$. For the time being, assume that $A - (K_d - 1)/2$ is positive, so the sense of inequality 2-14 remains unchanged when multiplying by this factor. If the square root is isolated on one side of the inequality, then

$$+ \sqrt{(A+\frac{S}{2}+J)^2-4J(A-\frac{K_d-1}{2})} \leq 2(A-\frac{K_d-1}{2})-A-\frac{S}{2}-J . \tag{2-15}$$

To reach this step, both sides have been multiplied by $A - (K_d - 1)/2$. Now, squaring both sides and collecting terms yields

$$2(S+K_d-1)(A-\frac{K_d-1}{2}) \leq 0 . \tag{2-16}$$

Since we assumed that the last factor was positive and since S is always a positive integer, equation 2-16 is contradicted. If $A - (K_d - 1)/2$

were assumed to be negative, equations 2-15 and 2-16 would still result,
but with the sense of the inequality reversed and inequality 2-16 would
again yield a contradiction. Thus the positive sign root in equation
2-13 is never of interest. A similar procedure also shows that the
minus-sign root always yields a value of T less than $1/\lambda$. Therefore,
the negative sign in equation 2-13 provides the root of interest.

Evidence regarding the accuracy of equation 2-13 under various
assumptions is summarized in Table 2-2. Equation 2-13, with $K_d = 1$,
provides an estimate for system throughput using Coffman's model for Q_d.
This estimate is compared with the simulation results to check its
accuracy. Since Q_p remains as an unknown, the value for Q_p is taken
from the simulation results. For the 195 simulation runs, the 90% point
error is 4.0% and the maximum error is 19%, compared to 6.5% when Q_d was
also taken from the simulation. Yet, the average error for the runs is
only 1.7%, less than twice the 0.98% error with Q_d directly from the
simulation. Note that this average error is the average of the
individual error magnitudes so positive and negative errors do not
offset each other. Even though Coffman's drum model assumes Poisson
arrivals, which is not the case in most of the data, a reasonably close
estimate of system throughput is obtained.

This particular model for T may also be compared under the
assumption that processor queue wait, Q_p, is negligible. In this way,
the model is only a function of the input parameters and no data is

Table 2-2.

Throughput Model Accuracy

(see equation 2-13)

Source of Q_d	Source of Q_p	T Error (in %)		
		Maximum	Average	90% Point
simulation	simulation	6.5	0.98	3.0
Coffman	simulation	19.0	1.7	4.0
Coffman	0	50.1	3.0	6.6
AGV	simulation	9.3	1.4	3.2
AGV	0	50.1	2.7	5.8

needed from the simulations. When Q_p is set to zero, the average error is 3.0%, the 90% point error is 6.6%, and the maximum disagreement between this model and the simulation is 50.1%. This maximum error case is one of only 14 which yield errors greater than 10%. In each of these cases, there is a significant amount of processor queue wait even with the processors occupied less than 90% of the time on the average. In order to have significant processor queue wait without having saturated processors, there must exist significant periods when the processors are all occupied and periods when some are idle. This phenomenon occurs only when the flow of jobs to the processor subsystem is very bursty(uneven). Of these 14 cases with high error, only one occurred when the simulation fault distribution was exponential. The remainder occurred with the much more bursty hyperexponential distribution. Also, only 3 high error cases occurred when the system had more than one processor, because the effective fault process is less bursty with multiple processors. If only the simulation cases with more than one processor are examined, the maximum error is 22.3%, the 90% error is 3.2%, and the average error is 1.6%. When the same model is used but Q_p is taken from the simulation, the cases with more than one processor average a 1.1% error with a maximum error of 8.3% and a 90% point error of 2.7%. Thus this model is quite accurate for most cases with more than one processor, even though processor queue wait is assumed to be negligible. As the number of processors increases, the flow of jobs through the system tends to exhibit a more even character, which more

accurately fits the model assumption of deterministic flow.

To examine the accuracy of the AGV model, K_d can be derived from the simulation data. When this evaluation is done, and also assuming that Q_p is zero, the average error for all the simulation runs is 2.7%, the 90% point error is 5.8%, and the maximum error is 50.1%. The maximum error is the same for Coffman's drum model, while the average for Coffman's model is 3.0%. Thus, there is a very slight improvement in overall accuracy. The maximum error remains unchanged because it is due to the error in processor queue wait. Very few of the cases are inaccurate because of poor drum queue wait estimation. In fact, Q_d is often only a very small part of the trip time, so even a large relative error for Q_d does not cause significant error in T. Those cases in which the AGV model cause improvement occur primarily for hyperexponential fault distributions, but even then, the improvement is slight. A few cases exhibit slightly greater error with AGV than with Coffman's model, but this effect is probably within the tolerance of the simulation error.

If the AGV model is used along with an accurate figure for processor queue wait taken from the simulations, the average error is reduced to 1.4%, the 90% point error is 3.2%, and the maximum error is 9.3%. Using Coffman's model these figures are 1.7%, 4.0%, and 19.0%, respectively. Again, the average error only improves slightly. However, the maximum error is halved. It should be noted that only in 5

out of 195 cases does the Coffman model yield an error greater than the maximum of 9.3% with the AGV model.

The model for Q_d in equation 2-11 requires a value for T. Earlier in this section, the value for T from the simulation was used to look at the accuracy of this model. After equation 2-11 has been incorporated into the deterministic model, however, another value for T is available. If T from the model is not equal to T from the simulation, then the deterministic model has implicitly used a different value for Q_d. Note that, in the region of interest, Q_d increases as T increases. Also, from equation 2-7, T decreases as Q_d increases. The combination of these two formulas in essence creates a situation of "negative feedback" which acts to reduce Q_d's extreme sensitivity to T described earlier. In particular, the prediction accuracy of the drum queue models can be compared to the simulation data when T from the deterministic model is used to compute Q_d. Under these conditions and setting Q_p to zero, both average and maximum errors are reduced. With Coffman's model, the average error is now only 8.7%, the 90% point error is 28.3%, and the maximum is 64.4%. The AGV model achieves an average error of 6.4%, a 90% point error of 28.3%, and a maximum of 51.3%. These values are listed in Table 2-3, along with the earlier data from Table 2-1. Virtually all of the improvement occurs in cases where Q_d is large and very sensitive to changes in T.

Table 2-3.

Drum Queue Model Accuracy

(see equation 2-11)

Model	Source of T	Source of Q_p	Q_d Error (in %)		
			Maximum	Average	90% Point
Coffman	simulation	–	360	13.7	33.2
AGV	simulation	–	322	10.5	24.0
Coffman	model*	0	64.4	8.7	28.3
Coffman	model*	simulation	64.4	8.3	19.0
AGV	model*	0	51.3	6.4	28.3
AGV	model*	simulation	51.3	5.8	19.0

* see equation 2-13.

Since Q_p was set to zero, it is possible that the model for Q_d might in fact be overestimating Q_d, thus partially covering for setting Q_p to zero. In fact, a comparison between Q_d from the simulation and the drum queue models using T from the model with Q_p provided by the simulation has even lower average error than before. The average error is now 8.3% for Coffman's model and 5.8% for the AGV model. The maximum and 90% point errors remained unchanged, and these results are also listed in Table 2-3. Also, in the system under study, it is unlikely that significant queue wait occurs at both drum and processor queues. When the system is processor bound, the drum queue wait will be quite small, and when the system is drum bound the processor wait will be negligible. Furthermore, when the value for Q_p from the simulation is included in the deterministic model, the model's prediction of throughput improves, as shown in Table 2-2.

Although the AGV model does provide some improvement in throughput prediction, it requires an estimate of the squared coefficient of variation of the drum arrivals. To date, no suitable estimate based on system parameters and throughput has been found. Therefore, the deterministic model utilizing Coffman's model for Q_d (equation 2-13 with $K_d = 1$ and using only the minus root) remains the only estimator of system throughput which is only dependent on the system parameters.

2.4 Processor Queue Wait Models

The primary source of error remaining in the analytic model is the lack of a model for processor queue wait. In this section potential models for the queue wait at the processors, Q_p are presented. In particular, two models from queueing theory are discussed. For the entire processor subsystem, a Markov model is derived which does not require explicit description of queue wait.

2.4.1 Queueing Theory Models

The primary source of error remaining in the analytic model is the lack of a model for processor queue wait. A simple model which might yield a useful estimate is the M/M/1 queue. This model, however, makes three assumptions which are not valid for the system under study: arrivals to the processor subsystem are assumed to be Poisson distributed, the service distributions are assumed to be exponential, and only one processor is assumed.

The queue waiting time for an M/M/1 queue is given by

$$Q_p = \frac{\lambda_p}{\mu_p(\mu_p - \lambda_p)} \quad , \tag{2-17}$$

where λ_p is the mean arrival rate to the processor subsystem and μ_p is the mean service rate of the processor subsystem. Since jobs are conserved in the system, the average flow of jobs leaving the processor

subsystem is equal to the average flow of jobs arriving. Therefore,

$$\lambda_p = \lambda_d = T\lambda \quad . \tag{2-18}$$

Now, μ_p must be related to the service rate of the processors. The simplest approach is to equate it to the aggregate service rate of the processor subsystem:

$$\mu_p = \frac{P}{\frac{1}{\lambda} + C} \quad . \tag{2-19}$$

Equation 2-19 has effectively replaced P processors with a single processor P times faster. Under these assumptions, substitution of equation 2-17 into the deterministic model, along with Coffman's model for Q_d (i.e., equation 2-12 with $K_d = 1$ and Q_p replaced), yields a third-order equation in throughput. Comparing the predictions of this model with the simulation data, a very poor fit is obtained. In particular, significantly better results occur when Q_p is set to zero, as described in section 2.2.3. The use of an M/M/1 model for processor queue wait overestimates the actual wait in almost every case. This overestimate is reasonable because in fact a single processor system of equivalent processing capability will have larger Q_p than a multiprocessor system. The multiprocessor system has zero processor queue wait until more than P jobs are in the processor subsystem, while the uniprocessor system has some queue wait with 2 or more jobs in the

42

subsystem. The best fits with the M/M/1 model occur for systems of one processor, as might be expected.

A reasonable extension of this approach is the use of an M/M/m model for processor queue wait, thus explicitly dealing with multiple servers. However, the expression for queue wait in such a system does not have a simple form [COFF73] and cannot be easily incorporated into the deterministic model.

2.4.2 Markov Model

In a system utilizing a paging drum, the drum model need only update its queues at intervals of one sector-time, which is also the basic unit of time in the model. Within each of these intervals it is only necessary to know the number of jobs which have arrived, ignoring their specific arrival times. Also, the drum can only return jobs to the processor subsystem at the end of each sector interval. So it is reasonable to consider modeling the flow of jobs departing the processor subsystem (and, thus, arriving at the drum) in a quantized fashion, based on the number of departing jobs per sector-time. In particular, such a model might be embedded in a simulation of the complete system to reduce the complexity of the simulator.

In the general case, the description of the processor subsystem at the start of a sector interval needs to include the number of jobs in the processor queue waiting for a processor, the number of processors,

the number of jobs currently running on processors, a descriptor of the faulting process, and the time remaining before each running job faults. A model which keeps track of each running job's time remaining to fault (which may extend over many sector-times) essentially stores all the information needed to fully simulate the system. A standard Markov model would be large and intractable due to its large state space.

If the distribution of faults is restricted to the exponential distribution (a common assumption) it becomes possible to consider each sector interval independently. The "memoryless" property of Poisson processes states that at any arbitrary point in time the time remaining before a job's next fault is independent of the past history of the job. Therefore, any job which is running on a processor at the end of the current sector interval may, in effect, be restarted at the start of the next interval. Now the model of the processor subsystem must deal only with the total number of jobs in the subsystem, the number of processors, and the fault descriptor which is now an exponential distribution with mean rate λ.

If the number of jobs in the subsystem were not limited, all processors would be continuously active and the number of jobs faulting would be described by a Poisson distribution. A_i, the probability that i jobs fault in one sector-time would be given by:

$$A_i = \frac{(\lambda P)^i}{i!} e^{-\lambda P} \qquad .$$

(2-20)

To find the actual probability distribution of the number of jobs faulting in one sector-time, it is necessary to consider the restriction imposed by the finite number of jobs in the subsystem, J_p. Let p_i be the probability that i jobs finish in one sector-time given J_p jobs in the processor subsystem at the start of the sector interval. This case may be related to the infinite job case by considering a subsystem in which there are an infinite number of jobs, but the first J_p jobs are "real" and all remaining jobs are "imaginary". The p_i's may then be computed using the Poisson distribution in combination with a discrete time Markov chain. During a sector interval, real jobs are allocated to processors until every real job has at least been started on some processor. After this point, imaginary jobs are allocated to processors. For the cases in which no imaginary jobs have been started on any processors, only real jobs can fault, so $p_i = A_i$ when i is less than or equal to $J_p - P$, since in this case only real jobs have been started. When i is greater than $J_p - P$, at least one imaginary job has been started and the value of A_i may include imaginary jobs. In order to compute the effects of imaginary jobs, a state diagram is useful(Figure 2-2). The states are labeled with the number of real jobs which have faulted. The number of state transitions required to reach a state along any possible path determines the total number of jobs, real and imaginary, which have faulted. The arc labels denote the probabilities of reaching the next state at the next fault. In other words, at state i the self loop describes the probability that the next

$$0 \xrightarrow{1} 1 \xrightarrow{1} 2 \xrightarrow{1} \cdots \xrightarrow{1} J_p - P$$

$$\downarrow 1$$

$$\tfrac{1}{P} \, \circlearrowleft \, J_p - P + 1$$

$$\downarrow \tfrac{P-1}{P}$$

$$\tfrac{2}{P} \, \circlearrowleft \, J_p - P + 2$$

$$\downarrow \tfrac{P-2}{P}$$

$$\tfrac{3}{P} \, \circlearrowleft \, J_p - P + 3$$

$$\downarrow \tfrac{P-3}{P}$$

$$\vdots$$

$$\downarrow \tfrac{2}{P}$$

$$\tfrac{P-1}{P} \, \circlearrowleft \, J_p - 1$$

$$\downarrow \tfrac{1}{P}$$

$$1 \, \circlearrowleft \, J_p$$
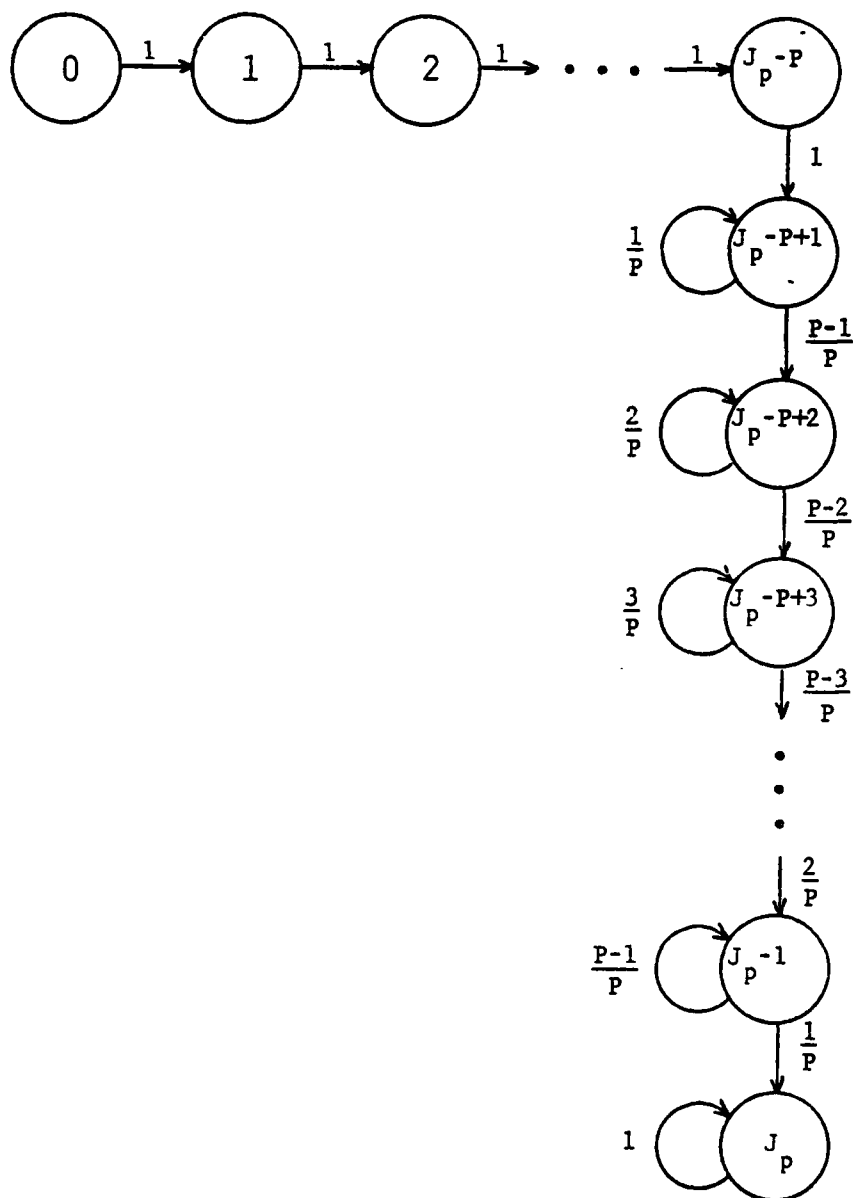
Figure 2-2. Markov Chain For Processor Subsystem.

fault is caused by an imaginary job. The transition arc to a new state is taken when a real job faults.

In the case of $i = J_p - P + 1$, one or more imaginary jobs may have faulted. By examining the state diagram, there are an infinite number of paths leading to the state $J_p - P + 1$. This effect occurs because once imaginary jobs have started any order of faulting of imaginary and real jobs is possible. The probability of i real jobs faulting is the sum of the probabilities of all paths leading to state i each multiplied by the Poisson probability of k jobs faulting, where k is the number of state transitions in that path. For $i = J_p - P + 1$,

$$p_i = A_{J_p - P+1} + \left(\frac{1}{P}\right) A_{J_p - P+2} + \left(\frac{1}{P}\right)^2 A_{J_p - P+3} + \cdots \quad . \quad (2\text{-}21)$$

In general, for $J_p - P < i <= J_p$,

$$p_i = \left[\prod_{k=1}^{i-(J_p-P)-1} \frac{P-k}{P} \right]\left[A_i + \left(\frac{1}{P}\right) \sum_{j_1=1}^{i-(J_p-P)} j_1\, A_{i+1} \right.$$

$$\left. + \left(\frac{1}{P}\right)^2 \sum_{j_1=1}^{i-(J_p-P)} (j_1 \sum_{j_2=j_1}^{i-(J_p-P)} (j_2\, A_{i+2})) + \cdots \right] \quad . \quad (2\text{-}22)$$

In this manner, values for each of the p_i's may be obtained, thus defining the job fault distribution for one sector-time.

For i greater than $J_p - P$, the value of p_i involves an infinite sum, and no closed form has been found. To find numerical values, it is necessary to terminate these infinite sums. It is possible to terminate the sums in such a way as to find both upper and lower bounds on the p_i's.

In order to obtain a lower bound, choose some integer n and define a new set of probabilities, A^l_i. Let $A^l_i = A_i$ for values of i less than n. Then, A^l_n is set equal to the sum of the A_i's for i greater than or equal to n, and the A^l_i's for all i greater than n are set to 0. The use of the A^l_i's will result in a distribution for a set of p^l_i's whose cumulative distribution is never below the cumulative distribution of the original p_i's. The distribution of p^l_i's thus overestimates the number of jobs faulting. As n is chosen larger, the bound becomes tighter.

An upper bound may be achieved in a similar manner. Define a set of A^u_i's such that $A^u_i = A_i$, for i between 0 and n. The A^u_i's for i greater than n are set to 0. Using equation 2-22 yields values for a set of p^u_i's. This assignment will effectively reduce the p^u_i's for $J_p - P < i < J_p$. In order for the p^u_i's to sum to one, set

$$p^u_{J_p} = 1 - \sum_{i=0}^{J_p-1} p^u_i \quad . \tag{2-23}$$

Thus, the probability of all J_p jobs faulting is increased and the

probability of from $J_p - P + 1$ through $J_p - 1$ jobs faulting is decreased. This assignment yields a cumulative distribution which is always less than or equal to the cumulative distribution for the p_i's, so it provides an overestimate, or upper bound, on the number of faulting jobs.

Unfortunately, there are several drawbacks to the use of this model. Since no closed form has been found, it would be necessary to calculate both the upper and lower bounds in order to verify the results. In addition, the model is less useful to describe the processor subsystem in a simulator than simply using random number sequences to generate the actual fault times of the individual jobs. To use the subsystem model in a simulator, tables of the p_i's would need to be generated and stored for all possible values of J_p, from 1 to J. Alternately, a single distribution of the p_i's could be calculated at each sector-time, but this would be very inefficient. Also, this subsystem model is valid only for exponentially distributed faults, while the use of a random number generator allows any distribution which can be derived from a uniform distribution to be used. For these reasons, the processor subsystem Markov model was not utilized in the simulator of the system. However, this model does provide a different perspective of the operation of the processor subsystem in the presence of exponential faults.

2.5 Final System Model

The deterministic model developed in section 2.2 is the basis of the final model. In particular, that deterministic model with Coffman's model for the drum subsystem is used in Chapter 3 to examine the performance of multiprocessor systems. The throughput of a system is given by

$$
T = \min\left\{ \frac{\frac{1}{\lambda}+C+\frac{S}{2}+1+J - \sqrt{(\frac{1}{\lambda}+C+\frac{S}{2}+1+J)^2 - 4J(\frac{1}{\lambda}+C+1)}}{2\lambda(\frac{1}{\lambda}+C+1)} , \frac{P\frac{1}{\lambda}}{\frac{1}{\lambda}+C} \right\} \quad (2\text{-}24)
$$

The first case is derived from equation 2-13 with $K_d = 1$ (Coffman's model), $Q_p = 0$ (no satisfactory processor queue delay model), and the negative sign selected. The second case directly corresponds to the processor saturated case of equation 2-7.

3. ANALYSIS OF RESULTS

Given the model developed in Chapter 2, one can easily examine the performance of multiple processor systems as a function of the system configuration and workload. In this chapter, the analytic model of throughput presented in equation 2-24 is used to look at several aspects of system performance. Where applicable, data from the simulations is provided for comparison.

First, the performance of the system as a function of each of the system parameters (P, S, J, C, and λ), individually, is discussed. Next, three interesting trade-offs are examined. The relationship between the number of jobs, J, and the fault rate, λ, is a factor in systems in which the jobs in the system must share a fixed amount of primary memory. Other trade-offs discussed involve the number of processors in the system. For instance, a system with many processors is compared to more conventional single processor systems. Finally, using P processors to achieve a P-fold performance improvement over a single processor is examined.

3.1 System Parameter Sensitivity

In this section, the graphical data presented describes a particular range of systems. The "base system" for this data is a system with 16 processors, a 32 sector drum, 32 jobs, an interfault time of 32 sectors (one drum revolution), and negligible processing overhead per fault. As each parameter is examined, the remainder of the parameters will usually remain fixed at their initial value, but some of the other parameters may take on multiple values to better illustrate system behavior. For instance, interfault time often takes on several values in order to show the range of system behavior from drum bound to processor bound.

First, the dependence on the number of jobs in the system is examined. Figure 3-1 shows some typical values of throughput for a wide range of J for several values of $1/\lambda$, the interfault time. It is generally desirable to operate a system with just enough jobs to keep the system at or near saturation. This crossover point is defined by equation 2-6. The system for the data in Figure 3-1 has 16 processors, 32 drum sectors, and negligible processing overhead per fault ($C = 0$). The interfault time, $1/\lambda$, ranges from 8 sector-times (one-fourth of a drum revolution) to 128 (4 revolutions). For high interfault times, such as $1/\lambda = 128$, the system is very nearly saturated with as few as one job/processor ($J = 16$). This phenomenon occurs for interfault times equal to 4 drum revolutions or greater. With moderate interfault times,
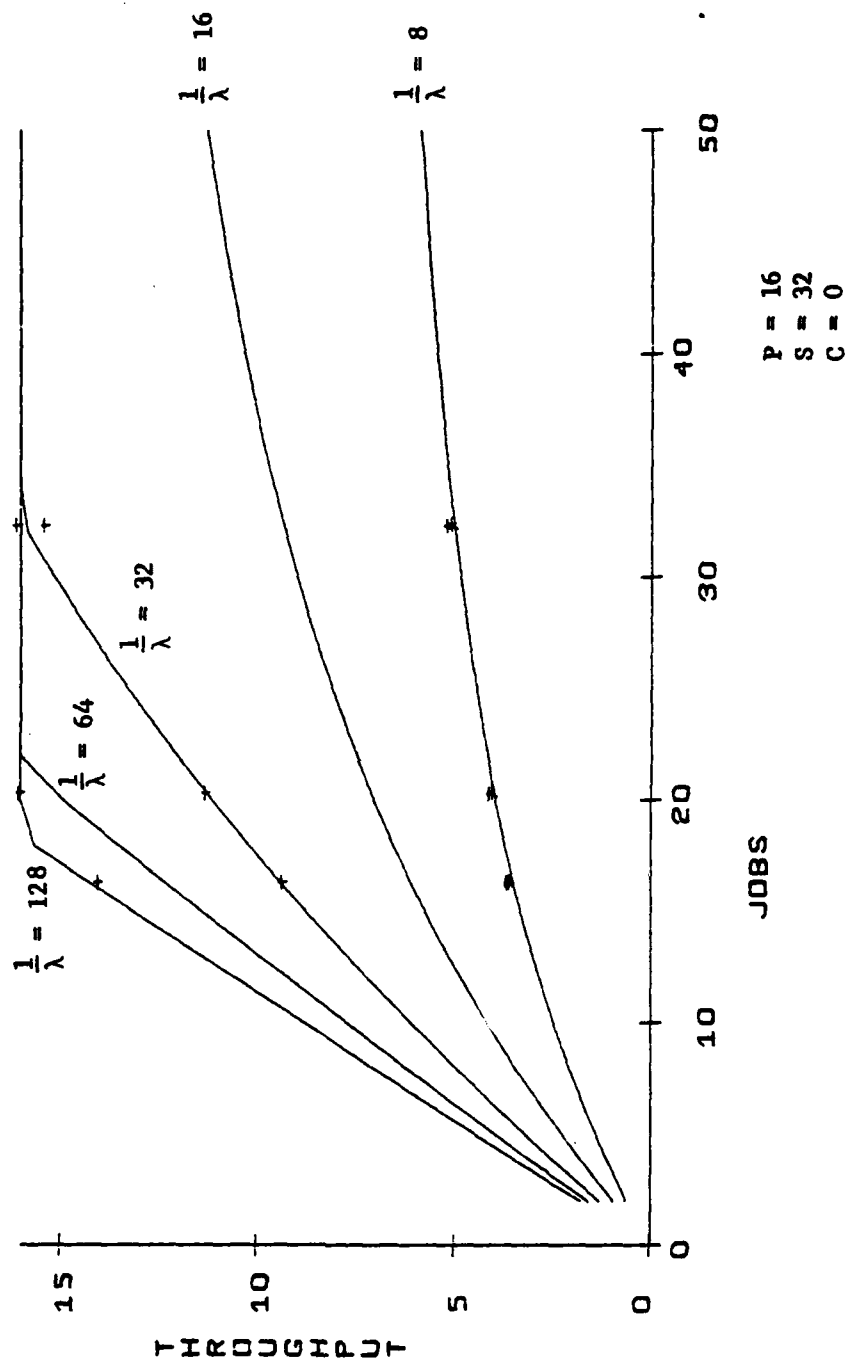
Figure 3-1. Throughput vs. J.

such as $1/\lambda = 32$, the system exhibits a nearly linear increase in throughput until around $J = 32$, or two jobs/processor. As interfault times continue to decrease the performance of the system begins to be limited by the drum. For instance, with $1/\lambda = 16$, it would take an infinite number of jobs to saturate the processors. This occurs because the service rate of the drum is 1 job/sector-time and the 16 processors request drum service at a mean rate of 1 job/sector-time only when fully utilized for this interfault time. For interfault times less than 16 the system is drum limited such that full processor utilization is not possible. Here, the system throughput is bounded by the service rate of the drum (also the arrival rate to the processor subsystem). The rates of job flow from the processor subsystem to the drum and from the drum to the processor subsystem must be equal. In general, throughput in the drum-limited case approaches an asymptotic value which is derived by setting the maximum flow rate of jobs leaving the processor subsystem equal to the drum service rate, or

$$\frac{P}{\frac{1}{\lambda}+C} = 1 \quad . \tag{3-1}$$

For example, with an interfault time of 8, a drum service rate of 1, and $C = 0$, the processor subsystem can deliver up to 1 job/sector-time with 8 processors. So drum throughput limits the throughput of this system to $T = 8$.

Also in Figure 3-1, the behavior of the system for fewer than 1 job/processor is shown. In this region the model is highly accurate, since the assumption of no waiting time in the processor queue is always true. Several discrete data points from the simulations are presented, and they agree with the model's prediction of performance quite closely (The "+" points correspond to exponential fault distributions and the "*" points to two-stage hyperexponential distributions with K = 16.). Even such a large change in K from 1 to 16 has little effect on system performance, suggesting that information about the second moment of the fault distribution may not be necessary.

The variation of T with changing interfault time, $1/\lambda$, is depicted in Figure 3-2. Again, this graph depicts P = 16, S = 32, and C = 0. $1/\lambda$ ranges from 1 sector-time to 128 (4 drum revolutions). Separate curves are drawn for 1, 2, and 3 jobs/processor (16, 32, and 48 jobs, respectively). With one job/processor, the throughput follows the interfault time in a smooth curve. Since there are no "extra" jobs in the system, there is no processor queue wait and any time a job spends at the drum causes one of the processors to be idle. Thus, as the ratio of processor service time, $1/\lambda$, increases with respect to drum service time, each job spends an increasing percentage of its time executing on a processor. This behavior continues in cases with more jobs in the system, except that now the "extra" jobs allow the processors to be occupied even when some of the jobs are being serviced by the drum. From Figure 3-2, as few as two jobs/processor allows the processors to
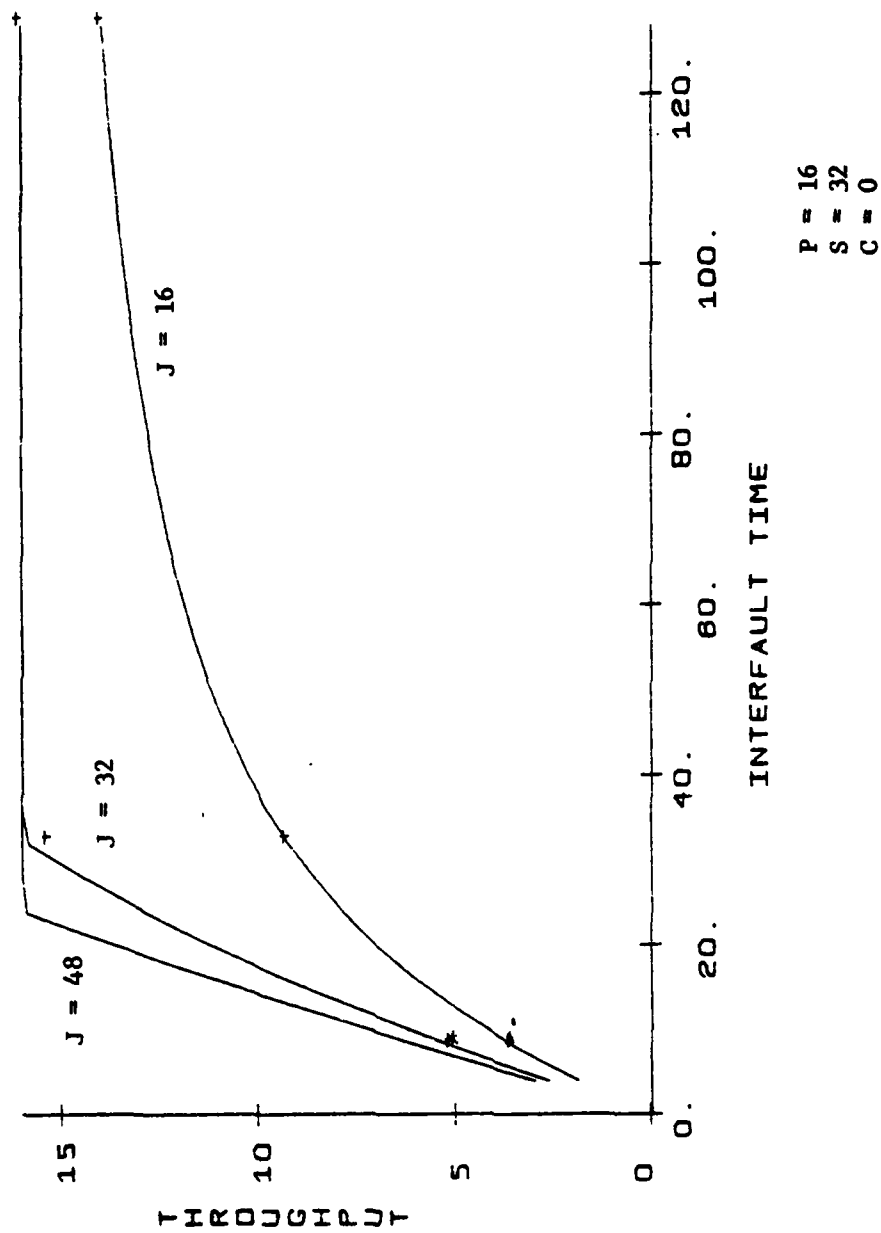
Figure 3-2. Throughput vs. $\frac{1}{\lambda}$.

be fully utilized when the interfault time is about 36, or slightly over one drum revolution. Additional jobs, such as three/processor, provide notably enhanced performance only in the range of interfault times of one-half to one revolution. The value of J which causes the processors to saturate is given in equation 2-6.

In Figure 3-3, the relationship between throughput and the number of processors, P, is shown. The solid lines correspond to systems with 16 jobs and dotted lines denote 32 job systems. In each case, two different interfault times are plotted. The diagonal line in the graph represents the processor saturated case, where $T = P$. Upon examination of equation 2-24, P only explicitly appears in the term for the processor-saturated case. In the processor-unsaturated term, the number of processors does not appear because Q_p, the processor queue wait time, is not specified in the model. For fixed J and $1/\lambda$, as P increases the throughput increases until the system is no longer processor bound. At this point, the system becomes bound by the number of jobs in the system, so the throughput is incapable of utilizing any additional processors and the curve is a straight horizontal line. If either J or $1/\lambda$ is increased, the processor saturated region is extended to a larger number of processors. To achieve high processor utilization, it is undesirable to operate too far to the right of the processor bound "diagonal line" in Figure 3-3. This intersection point between saturation and unsaturation may be obtained by resolving equation 2-6 for P. The processors are saturated when

57
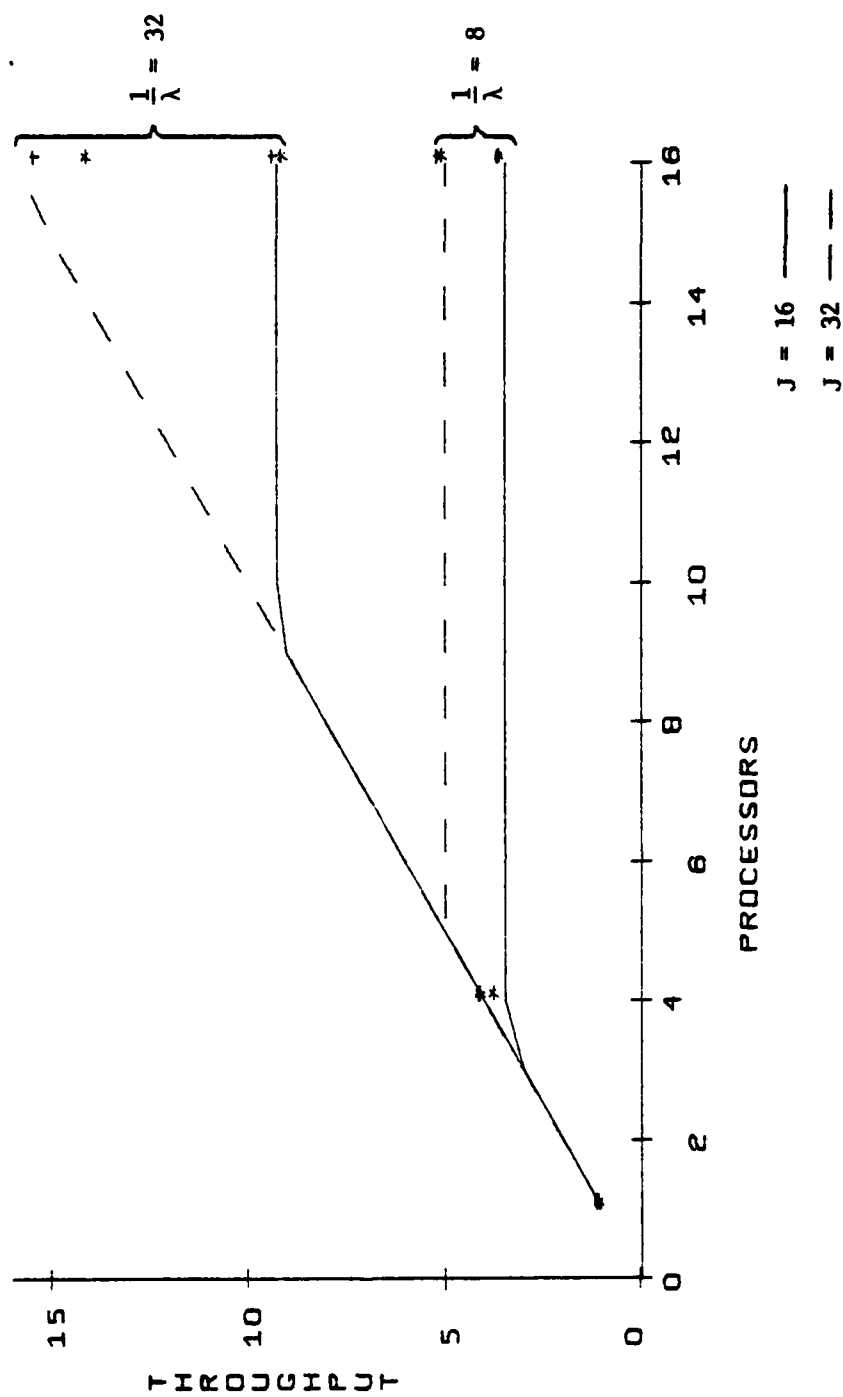
Figure 3-3. Throughput vs. P.

$$P \leq \frac{J(\frac{1}{\lambda}+C)}{\frac{1}{\lambda}+C+1+Q_d} \, . \qquad (3\text{-}2)$$

The variation of throughput with a changing number of drum sectors, S, is depicted in Figure 3-4. For this graph, both $1/\lambda$ and C are expressed in sector-times so as S changes they remain unchanged. However, the revolution time of the drum changes linearly with S so processor execution will not remain constant with respect to the revolution time of the drum. As might be expected, the throughput tends to decrease with increasing S, because of the additional rotational delay for each service. The potential decrease in drum queue delay brought about by spreading the arriving jobs among more queues is negated by the fact that drum rotation time also increases linearly with S. As S tends to infinity, the throughput approaches zero, since the drum becomes the bottleneck of the system and Q_d goes asymptotically to S/2. The top curve in Figure 3-4 remains at T = 16 only because the particular values of J and $1/\lambda$ require S > 128 before throughput begins to degrade. The crossing of the two curves occurs because in one case $(1/\lambda = 32)$ the throughput is more dependent on drum performance whereas the case with a much larger interfault time (128) is mostly limited by its number of jobs.

When both $1/\lambda$ and C are expressed as constant multiples of a drum revolution time, S, the performance of the system with changing S is shown in Figure 3-5. Here, a change in S represents a change in density
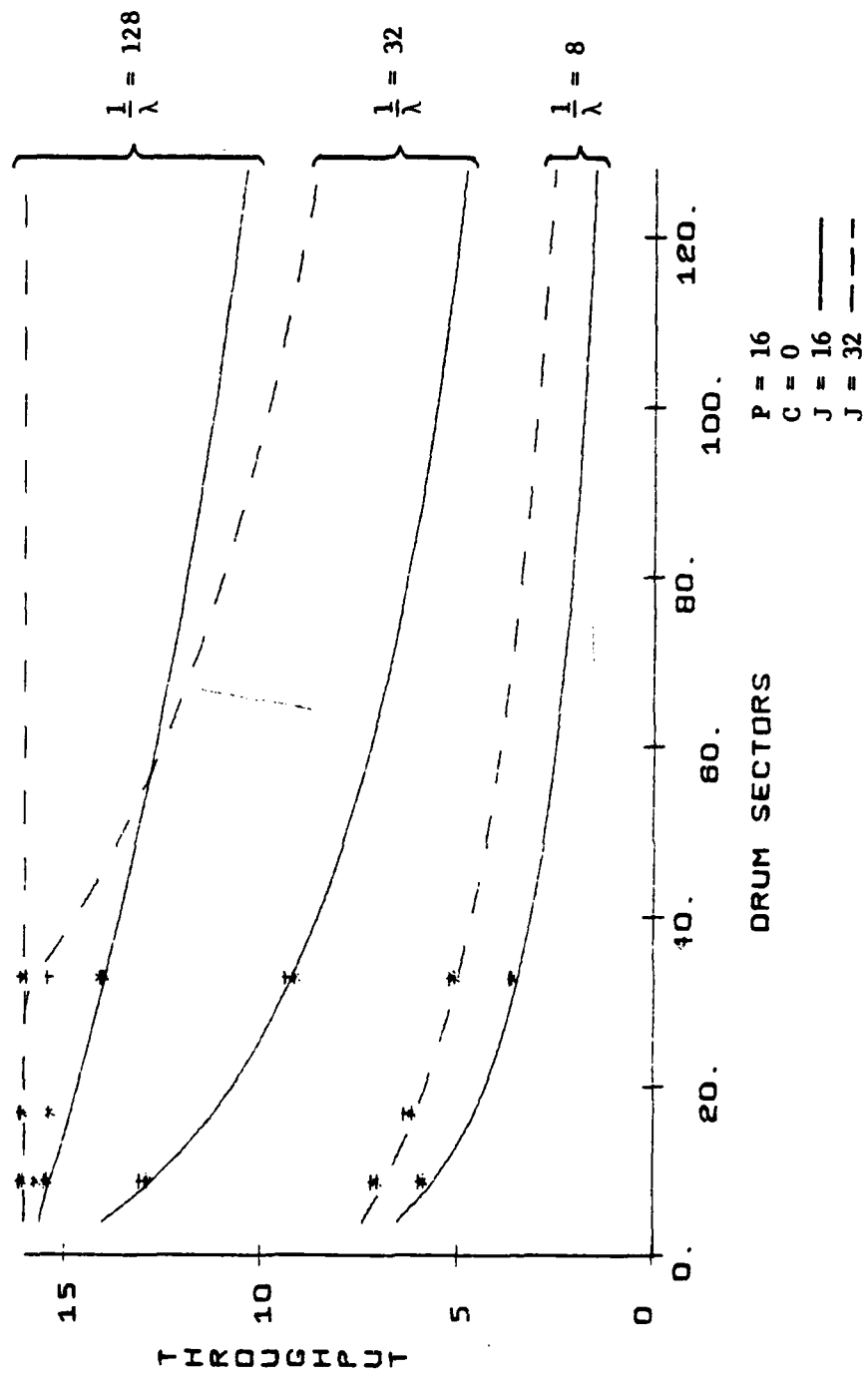
Figure 3-4. Throughput vs. S.
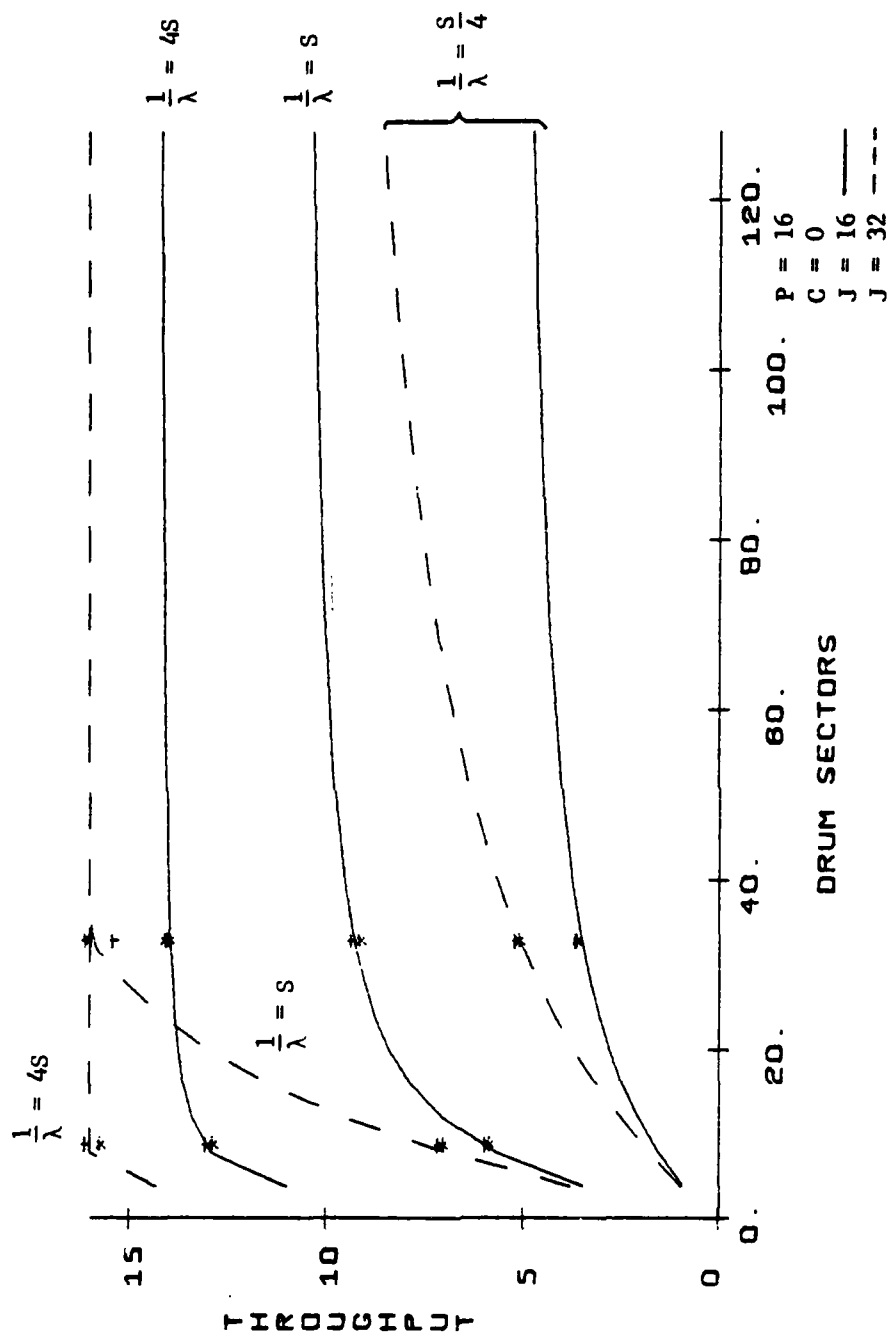
Figure 3-5. Throughput vs. S, Constant Faults/Revolution.

on the drum without affecting the rotational speed. As S increases, the queueing delay caused by waiting for jobs which arrived previously decreases because the jobs are spread over a larger number of sector queues. Now, however, the average of one-half rotation of the drum remains constant. This effect accounts for the general flattening of the curves for large S. In addition, the service time at the drum decreases with increasing S since S sectors can be read each revolution. This benefit is most pronounced when the system is more drum-bound, as in the curves for $1/\lambda = S/4$. When S is very small, both the service time at the drum and the queueing delay caused by other jobs increases. The queueing delay increase is also most noticeable with a large number of jobs.

The effects of non-zero processing overhead per fault, C, are illustrated in Figure 3-6. Increases in C can only reduce system performance. When the interfault time is low $(1/\lambda = 8)$ the system is bound by the drum and changes in processor execution times have very little effect on throughput, even when the processors are spending much greater time in overhead than in doing useful work $(C \gg 1/\lambda)$. For larger $1/\lambda$, though, performance can suffer considerably as C increases. For $1/\lambda = 32$ and $J = 32$, throughput is cut by almost half when C is equal to $1/\lambda$. When the interfault time is increased to 128 the throughput appears to suffer less as C increases, and this phenomenon is simply due to the fact that for the data shown C is only one-fourth of $1/\lambda$, at the most. Since the system is more processor limited as
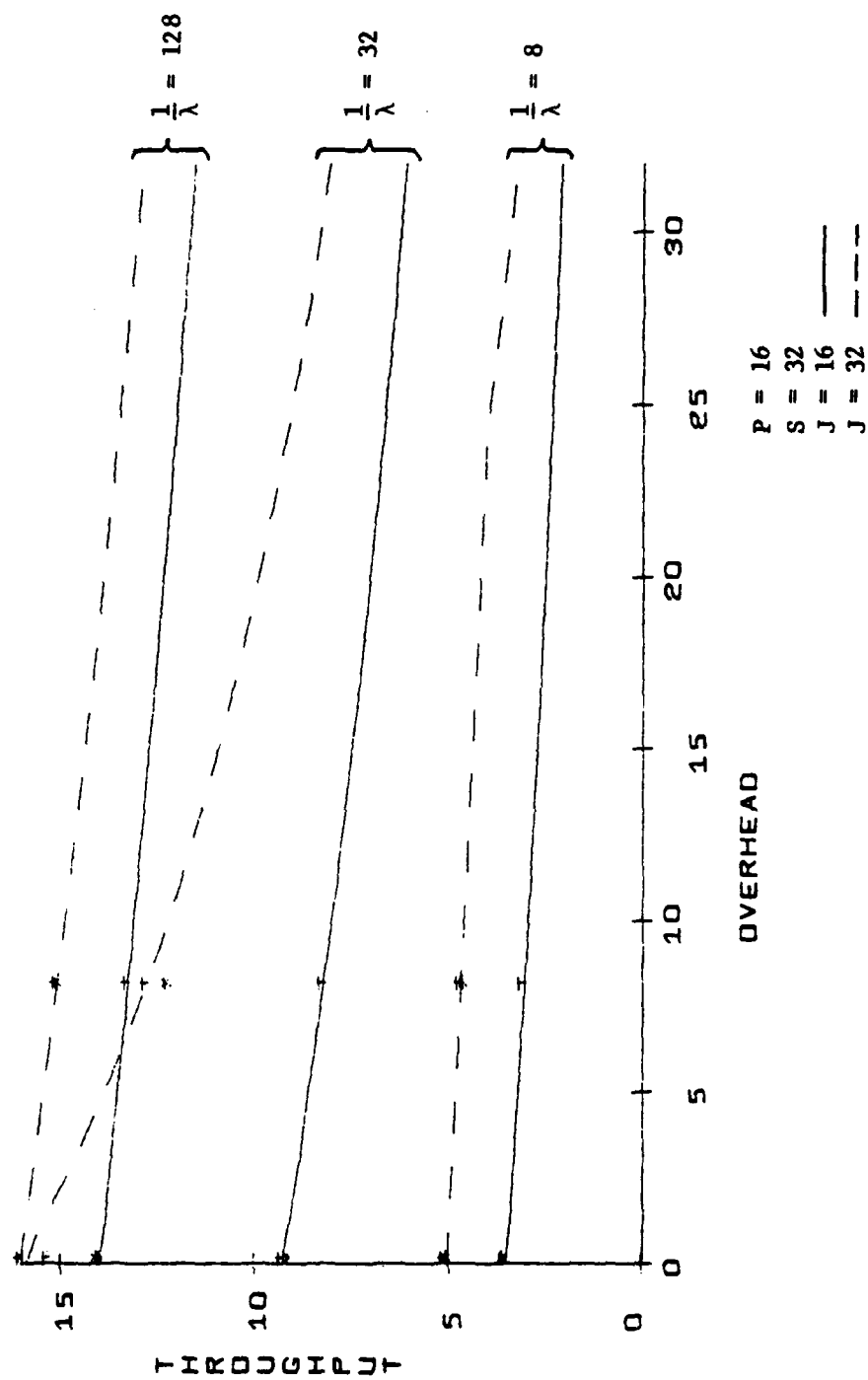
Figure 3-6. Throughput vs. C.

interfault time increases, the percentage degradation in performance for a given ratio of C to $1/\lambda$ increases with increasing $1/\lambda$.

Subsequent sections of this chapter deal with trade-offs in system performance.

3.2 J vs. λ Trade-offs

In this section, the relationship between the number of jobs in the system, J, and the fault rate of each job, λ, is examined. This relationship is especially useful when discussing the effects of memory size and memory allocation per job. As mentioned in Chapter 1, the dependence of fault rate upon memory size and allocation is specified by an as yet undetermined fault model. Since this research only deals with the system model, the details and effects of memory allocation and size are not known. Even with this restriction, it is possible to describe some meaningful dependencies related to memory. In particular, the trade-off between interfault time, $1/\lambda$, and jobs, J, can be shown for constant values of throughput, T. Thus, in order to maintain constant T when either J or λ is changed, the necessary change in the other parameter can be ascertained. For a given system, the values of P, S, C, and M (memory size) may be already chosen. Increasing the number of jobs in the system will decrease the memory allocation assigned to each job.

In the processor-unsaturated region, equation 2-12 with $Q_p = 0$ and $K_d = 1$ may be rearranged for λ, yielding

$$(C+1)T^2\lambda^2 + (T^2 - (C + \frac{S}{2} + 1 + J)T)\lambda + J - T = 0 \quad . \tag{3-3}$$

This second-order equation in λ has two roots, but as before, one root implies that Q_d is negative. The remaining root is

$$\lambda = \frac{C + \frac{S}{2} + 1 + J - T - \sqrt{(T - (C + \frac{S}{2} + 1 + J))^2 - 4(J - T)(C + 1)}}{2(C+1)T} \quad . \tag{3-4}$$

Now, equation 3-4 may be graphed for several values of T while varying J. Figure 3-7 shows this relationship in a system with P = 16 and S = 32. Two values of C are used: 0 (solid lines) and 32 (dashed lines). For the case of non-zero overhead per fault, C = 32, it is possible for the processors to become saturated. The point at which this relationship occurs was defined in equation 2-6. For values of J greater than this saturation point, it is impossible to achieve the specified value of T with a smaller value of interfault time. Thus, the dashed curves in Figure 3-7 become horizontal lines to the right of the point of saturation.

For both C = 0 and C = 32, there are two general regions in Figure 3-7. When the number of jobs in the system is small, the addition of even a single job greatly reduces the interfault time which achieves the same value of T. In this region, then, system throughput will generally

65

JOBS

$\frac{1}{\lambda}$

P = 16
S = 32
C = 0
C = 32

T = 12

T = 16
T = 8
T = 12
T = 8
T = 4

INTER FAULT TIME

100.

50.

0.

0   5   10   15   20   25   30
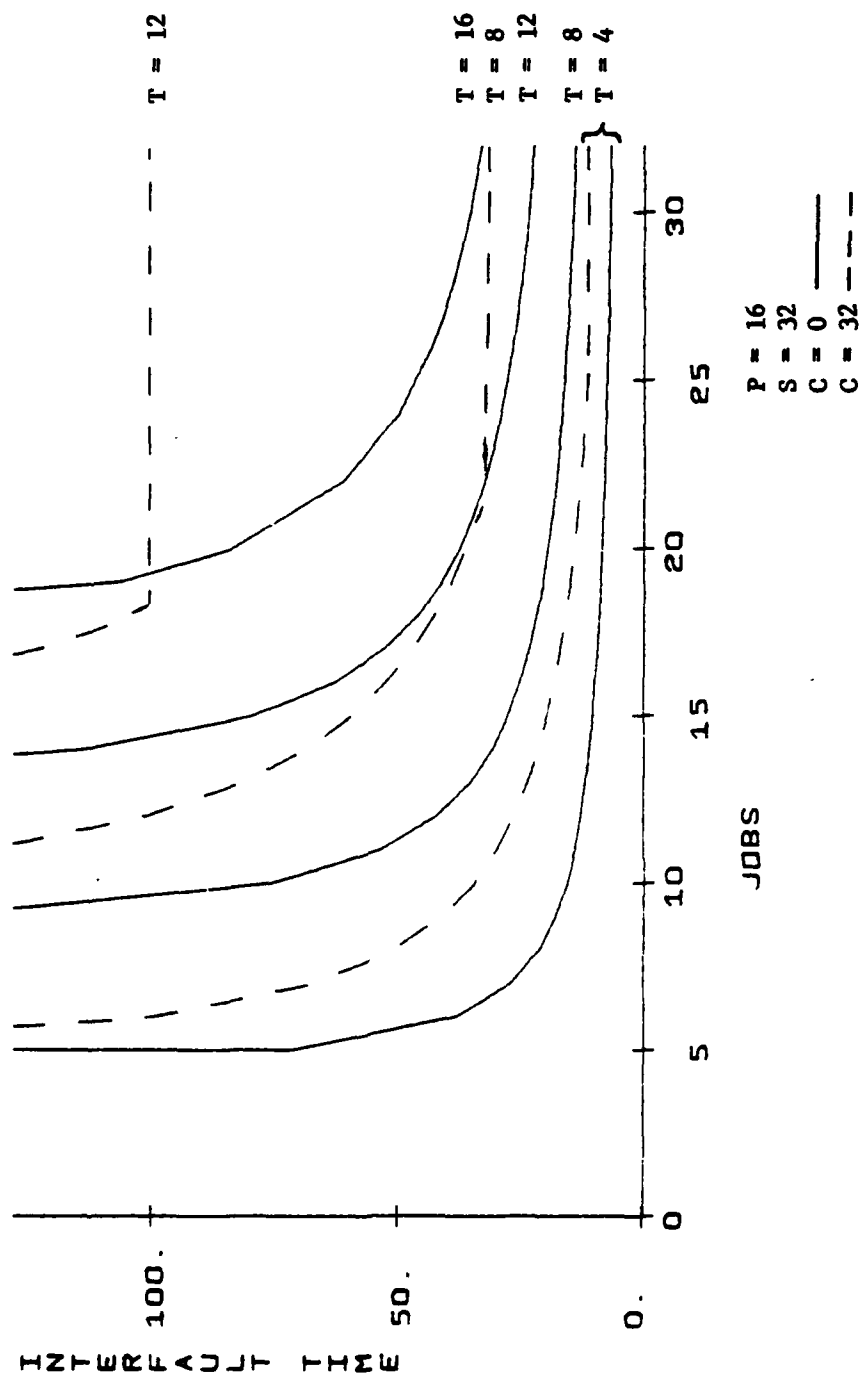
Figure 3-7. $\frac{1}{\lambda}$ vs. J for Constant T.

increase when jobs are added to the system, unless interfault time
decreases even faster than shown in the graph. When the number of jobs
is large, interfault time must only decrease slightly as jobs are added,
if the value of system throughput is to be maintained. Between these
two regions, the system undergoes a transition from one region to the
other, in which system throughput is sensitive to changes in either
parameter.

Although the relationship between memory allocation and fault rate
is not known, an example of the performance trade-off with an assumed
relationship might prove informative. For simplicity, assume that the
fault rate of a job is linearly related to its memory allocation. In
other words, if a job's allocation is doubled, its fault rate will
halve. Now interfault time can be expressed as a function of the number
of jobs in the system, assuming that the jobs share a fixed memory space
equally. If J doubles, the fault rate of each job, λ , will double.
Using equation 2-24, fault rate λ is now set to nJ, where n is a
constant. Figure 3-8 shows the dependence of throughput on J for three
values of n (0.00391, 0.000977, and 0.000244) and for two values of C (0
and 32). It is obvious that under the current assumptions, there is
often an optimum value for J. When C = 0 (solid lines) the processors
are not saturated until T = 16, and for two values of n this value of T
may be obtained. However, when n is large, corresponding to a higher
fault rate for a given J, the system throughput is bounded by the
operation of the drum and T decreases for increasing J greater than 15.
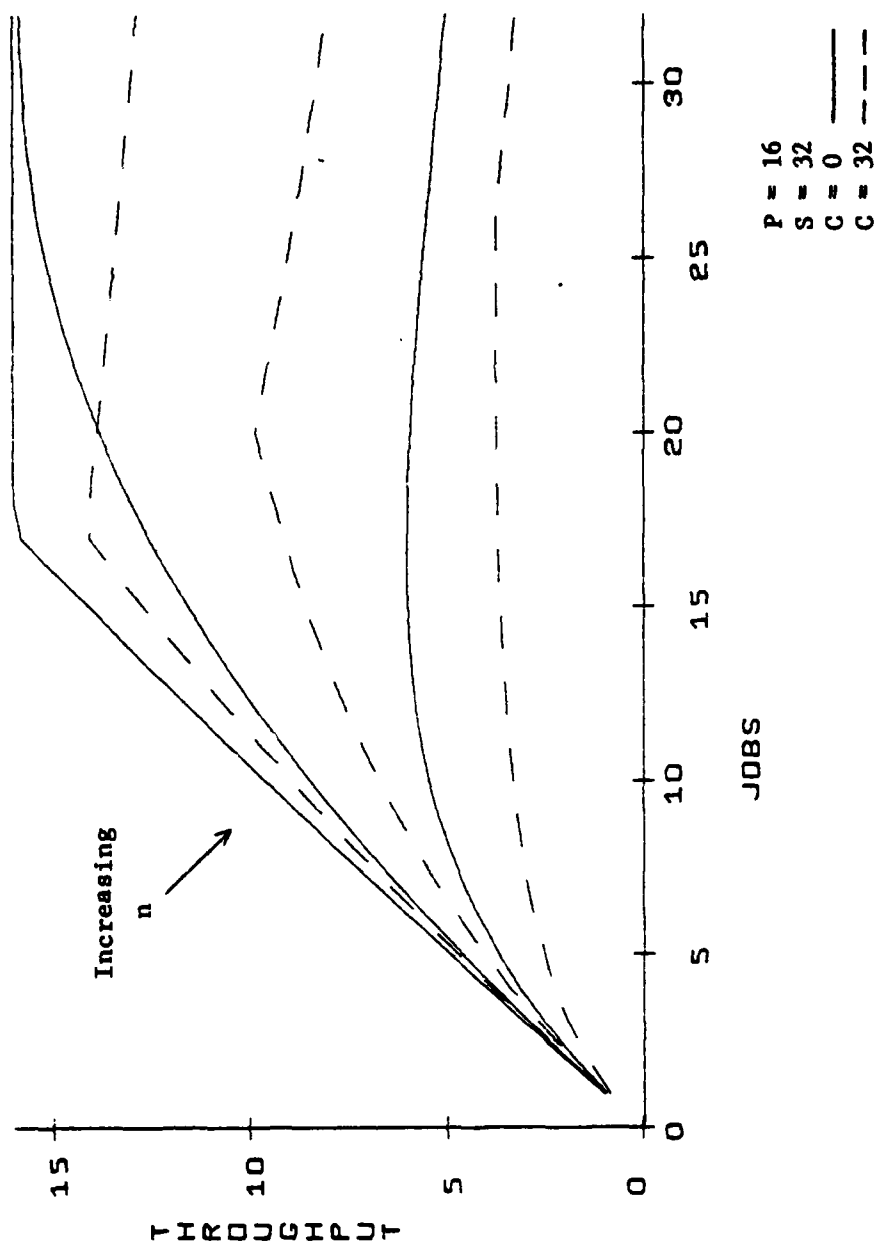
Figure 3-8. Throughput vs. J, with λ = nJ.

When C = 32 (dashed lines) and n is large the same drum bounded effect occurs but at a lower value of T, since some processor time is now spent doing overhead. For somewhat lower values of n, though, processor saturation becomes the limiting factor. When the processors become saturated, adding jobs does cannot make the processor more busy, and each job's fault rate increases since it has a smaller memory allocation. It is this increase in fault rate which causes T to decrease.

The above example uses a simple linear relationship between J and λ. Although the actual relationship is probably much more complex, the maximum attainable throughput for a system with fixed memory size can be ascertained in a similar fashion.

3.3. Processor Trade-offs

In considering the processor section of a multiprocessor system, questions arise concerning the performance of multiprocessor systems relative to a conventional single processor system. First, how might the throughput of a P-processor system compare to the same system with just one processor which is P times faster than each of the multiple processors? Second, how does the performance of P independent single processor systems relate to one combined P-processor system?

The first question, which deals with the replacement of a single fast processor by many slower processors, may be viewed as replacing a conventional large computer's processor with P tightly coupled mini- or micro-processors. The speed of a processor is reflected in the execution time of a job running on that processor. If a job running on a processor executes for a period of $1/\lambda + C$, then it will execute for a period of $P(1/\lambda + C)$ on a processor which is a factor of P slower. Throughput, as defined in Chapter 1, is expressed as the average number of processors doing useful work. When comparing systems whose processors have different speeds, this definition of throughput is not meaningful. A measure of system performance which is independent of processor speed is the _flow_ _rate_ of jobs leaving the processor subsystem, expressed in jobs per drum sector-time. For a given system, this flow rate is obtained by multiplying the throughput, T, by the fault rate of a job, λ .

Examination of the model for throughput in equation 2-24 reveals an interesting point: the number of processors in the system is only explicitly included in the processor saturated case. Here, the replacement of a single fast processor with P slow ones has no effect, because as the number of processors increases by a factor of P the fault rate of each processor decreases by a factor of P. Thus the maximum achievable flow rate is the same in both systems. When the processors are not saturated, the model assumes that throughput is limited by the number of jobs in the system and not by the number of processors.

However, the throughput in this region is directly affected by the speed of the processors. For instance, if a processor is half as fast as before, the execution time of a job on this processor must double. The interfault time becomes $2/\lambda$ and the overhead time becomes $2C$.

Figure 3-9 depicts the <u>relative flow rate</u> for various numbers of processors. Relative flow rate is defined as the flow rate of a P-processor system divided by the flow rate of a uniprocessor system with equivalent total processor capacity. Note that this relative flow rate never exceeds one, so the uniprocessor system always outperforms the multiprocessor version. When the systems are lightly loaded, at any point in time the jobs are partitioned between the processor subsystem and the drum subsystem. At any point in time, the number of jobs in the processor subsystem dictates the number of busy processors. For the uniprocessor system, only one job needs to be in the processor subsystem to keep it totally busy. On the other hand, with P processors at least P jobs must be in the processor subsystem for full utilization. This effect accounts for the lower performance of the multiple processor systems. A similar observation is also mentioned in a paper discussing one and two processor systems by Sauer and Chandy [SAUE79]. The solid lines in Figure 3-9 correspond to a system with only one job and with a range of interfault times. Here, the multiprocessor systems are at a severe disadvantage because at most one processor can be busy at any time. Note that the actual performance levels for these curves is quite low. The dashed lines describe the same range of interfault times with
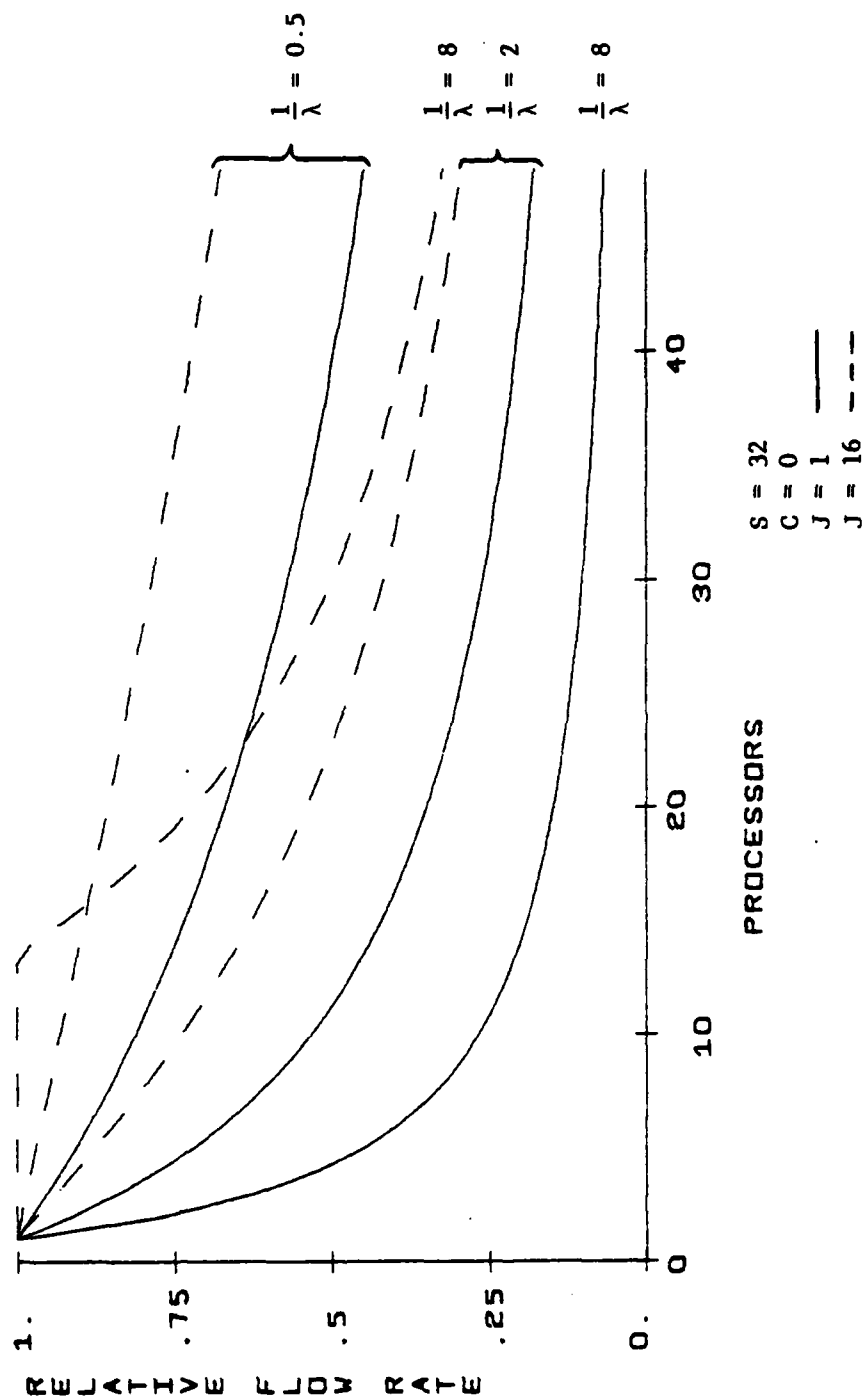
Figure 3-9. Relative Flow Rate vs. P.

16 jobs in the system. An interesting phenomenon occurs here. Generally, as interfault times increase the relative performance of the multiprocessor systems degrades. This effect occurs because the performance of both systems is more dependent on the processor subsystem as the interfault time increases. However, when the interfault time is large enough to cause both systems to become processor saturated, the relative flow rate is one, as depicted by the top line in the graph. As the number of processors increases to the point where the multiprocessor system comes out of saturation, its performance relative to the uniprocessor system shrinks rapidly, crossing the lines for lower interfault rates. Remember that these curves are only relative performance levels, and the crossing of different curves does not imply that the actual levels of performance are the same. The range of $1/\lambda$ used here is lower than that used in previous sections because if $1/\lambda$ is larger or if J/P is larger the one processor systems stays saturated and the relative performance is just a function of the multiprocessor system. In general, then, the multiprocessor approach is closest in performance to the uniprocessor system when the number of jobs is large and the system is operating at or near processor saturation.

The second trade-off discussed in this section is the comparison of P independent single processor systems with one P-processor system. In this case the total available processor capacity is the same, but one tightly-coupled multiprocessor system is contrasted with many independent uniprocessor systems. Such conglomerations of systems often

exist at computer centers which need greater performance than can be obtained by a single large uniprocessor system. Assume that each uniprocessor system has a drum with S sectors, J jobs, and each job has an interfault time of $1/\lambda$ and overhead time of C. For the combined system there will be PJ jobs. However, the choice of the drum subsystem for the combined system affects the interfault time and the overhead time as well as the number of sectors, because the unit of time in the model is one sector-time.

There are three candidates for the drum subsystem of the combined system. The most obvious choice is P drums identical to each drum in the uniprocessor systems. Unfortunately, the model from Chapter 2 only describes systems with a single drum. A close approximation of the P drums may be made by using a single drum with P times as many sectors but a rotation time the same as each individual drum. In this way, every drum rotation time PS sectors can be read and transferred, as with the P independent systems. An alternative interpretation of this system is a single drum with a P-fold increase in density which can actually support PS sectors. With this drum model, it is now possible to choose values for interfault time and overhead time. The drum revolution time is now PS sector-times. A job which executed for one revolution in one of the individual systems should still run for one revolution in the new system. Thus, the interfault time in the new system should be P times greater than that for a uniprocessor system, or P/λ. Similarly, overhead time should be scaled to PC. With these values it is possible

to compare the throughput of the P uniprocessor systems with the new system, using equation 2-24. When this comparison is done for up to 46 systems and wide ranges of the other parameters, the throughput is virtually identical. The combined system does achieve as much as a 5% improvement, simply due to slightly lower drum queue wait.

Unfortunately, the model does not include a term for processor queue wait, and the primary benefit of the P-processor system is the ability to share the total job load in a single processor queue. This effect would be most pronounced when the systems are near saturation and there is significant processor queue wait. When the systems are either lightly loaded or saturated the combined system should perform essentially the same as the aggregate of the independent systems.

Another choice for the drum subsystem is the use of a single drum identical to one in a uniprocessor system. Since the combined system now has fewer resources (i.e., only one drum instead of P), it should not perform as well as the P independent systems, particularly when throughput is limited by the drum. The relative throughput, computed as the throughput of the combined system divided by P times the throughput of a uniprocessor system, is shown in Figure 3-10. The solid lines correspond to one job per processor and the dashed lines to two, for various values of fault rates. As the job load (J/P) increases, the combined system degrades further because the drum subsystem is becoming the limiting factor.
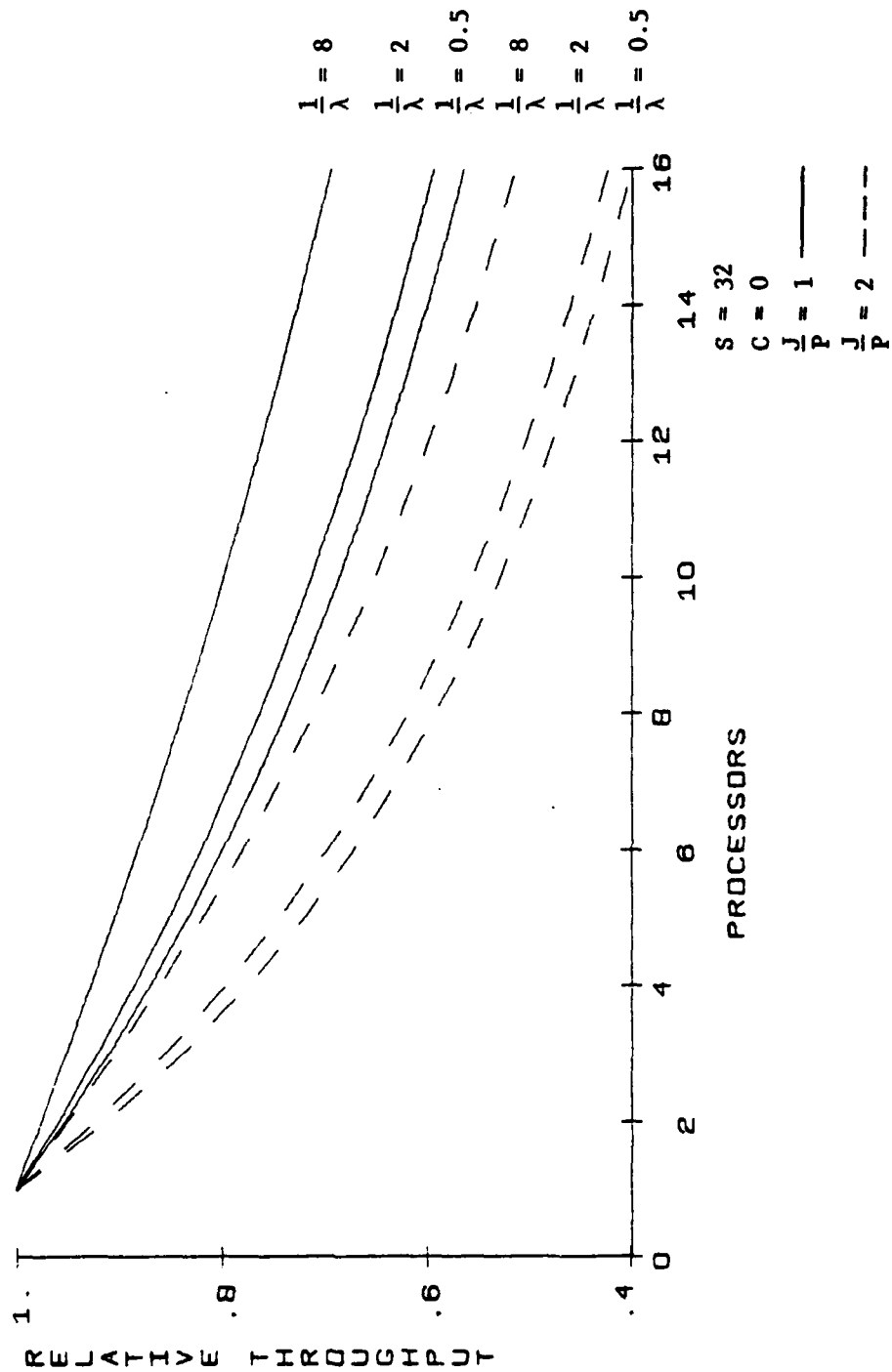
Figure 3-10. Relative Throughput vs. P.

A third drum subsystem for the combined system is a single drum which is capable of rotating P times faster than a uniprocessor's drum. If these two drums have the same number of sectors, then the maximum transfer rate of one fast drum and P slow ones is identical. However, the delay that a single job sees is quite different. Since P times as many jobs share the same number of sectors, the drum queue delay may increase. On the other hand, the rotational latency that each job confronts is now only $1/P$ times that of the original drum. The relative throughput of these systems is graphed in Figure 3-11 for the same range of parameters used in the previous graph. Here the combined system is always superior to the independent systems, and it compares most favorably under light loads as before. As the system load increases and the processors approach saturation, the throughput becomes dependent upon processing power, which is identical.

When comparing independent uniprocessor systems to a single multiple processor system with equivalent processing power, a choice of drum configuration must be made. The relative performance of the combined multiprocessor system depends heavily on this choice. If the drum of the combined system is either significantly faster or slower than the total of the drums in the independent systems, the combined system will perform either better or worse, respectively. It is difficult to comment when the drums are essentially the same, because the relative performance depends primarily upon processor queue wait, which is not explicitly included in the model.
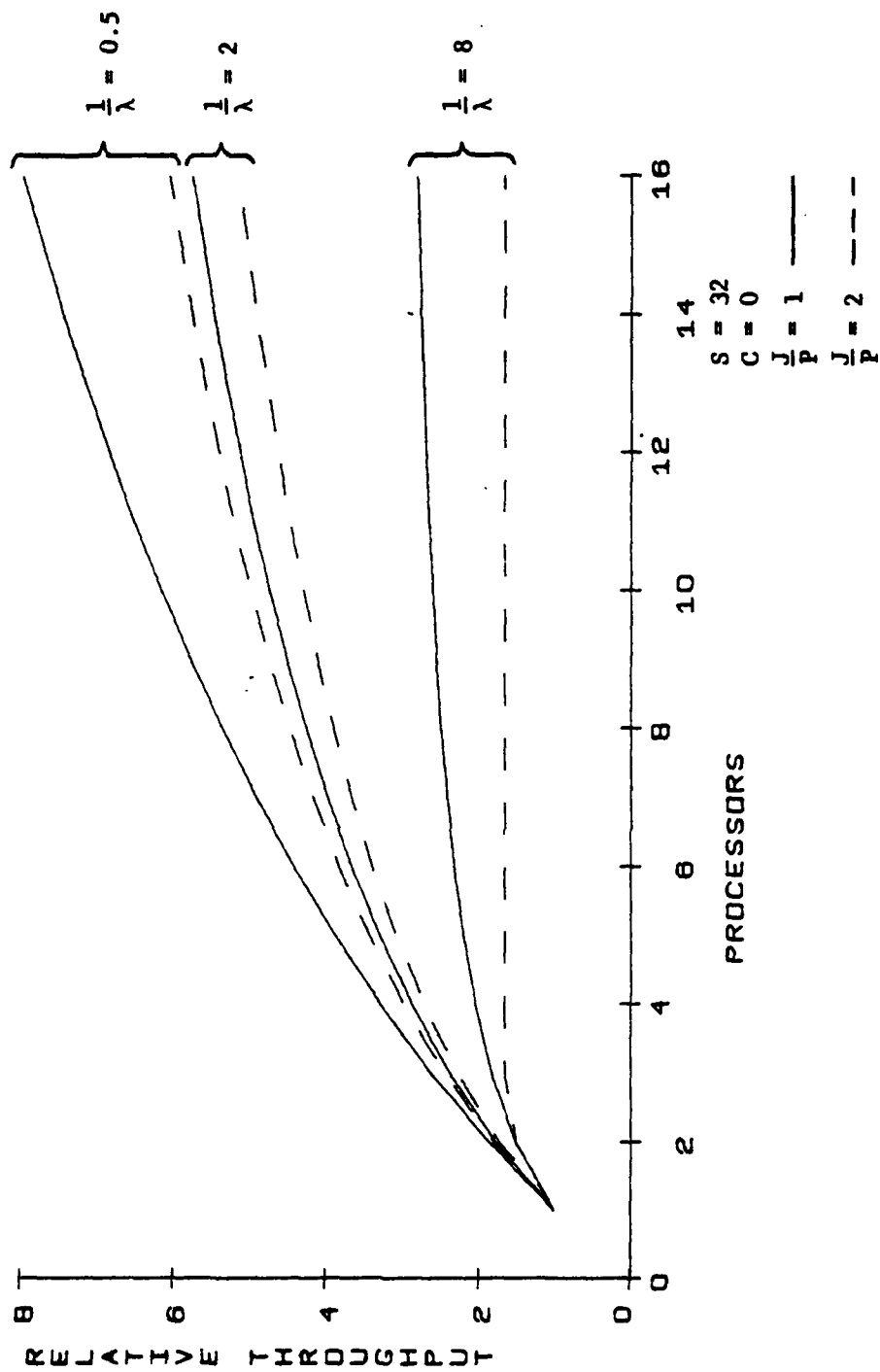
Figure 3-11. Relative Throughput vs. P.

4. CONCLUSION

4.1 Summary

This research has been concerned with performance modeling of multiprocessor computer systems utilizing a shared drum secondary memory. Interest in such systems is increasing because of the advancement of integrated circuit technology. Large-scale and very-large-scale integration have very high cost performance ratios, but have not been able to achieve high performance. In order to benefit from these high cost performance ratios more complex architectures are needed to increase the performance level of integrated circuits.

The system under study contains a fixed number of jobs. The P identical processors share a single job queue, and any job may be started on any processor. A job executes on a processor until it faults (or makes an I/O request). It then continues to execute on that processor for the amount of time necessary for the operating system to handle the fault. The job then travels to the drum subsystem. This subsystem consists of a paging drum with one queue per sector in order to employ shortest latency time first (SLTF) scheduling. After the job's page request has been satisfied, the job returns to the processor

queue to await an available processor.

The combined system model was serially decomposed into two models: a fault model and a system model. The faulting behavior of the jobs is described by a fault model. This model utilizes a program's paging behavior and the configuration of the memory to produce a fault descriptor. The system model uses this fault descriptor and the configuration of the physical system to predict system performance. Decomposition in this manner allows an examination of system performance without knowing the details of program paging behavior. The remainder of the research dealt with this system performance model.

The primary result of this research is the development of an analytic model for system throughput. Utilizing both deterministic scheduling and previously known queueing models, the system model expresses throughput as a simple formula based on six parameters: the number of processors, P; the number of jobs, J; the number of drum sectors, S; the operating system overhead per fault, C; the mean of the interfault time distribution, $1/\lambda$; and the squared coefficient of the interfault time distribution, K. Throughput, T, is the average number of processors in the system doing useful work. Although this deterministic model makes several simplifying assumptions about system operation, it agrees quite closely with an extensive number of simulations which cover a wide range of values for the parameters. In particular, the average relative error of the model when compared to all

195 simulations is only 3.0%. The model's error is just 1.6% when only the cases with more than one processor are considered. Other such simple models have been found to work very well, as in the Tailor model developed by Blake [BLAK79].

The accuracy of the model would indicate that perhaps only the mean of the interfault distribution is needed. The accuracies cited were achieved with a model using only the mean of the interfault distribution. When a sub-model for the drum system was examined which utilized the second moment of job flow in the system, only very slight improvements were obtained (an average relative error of 2.7%). All of these error figures include simulation runs whose squared coefficient of variation of the fault distribution ranged from 1 (exponential distribution) to 16 (two-stage hyperexponential).

The analytic system model was used to examine aspects of system performance. First, the dependence of system throughput upon each of the parameters individually was presented. In this way, the effects of variations in any one of the parameters can be predicted.

The relationship between the number of jobs and each job's fault rate was discussed, because when the jobs share a fixed amount of memory the allocation per job must change as the number of jobs changes. In particular, the trade-off between J and λ was detailed for constant values of throughput. By using these curves of constant performance, the decision to add or subtract jobs in an existing system can be

reduced to a simple binary choice. For instance, if the fault rate of the real system increases with the addition of a job more than the increase in fault rate specified on the appropriate constant performance curve, then the addition of that job will reduce system performance.

Several trade-offs dealing with the number of processors in the system were examined. These trade-offs specifically dealt with comparisons between various multiprocessor systems and conventional uniprocessor systems. First, a system with P processors was compared to the same system with a single processor which executes P times faster than one of the multiple processors. Although the potential throughput of both systems is identical, it was found that when the systems are not saturated, or constantly busy, the single processor system exceeds the performance of the multiprocessor system.

Another trade-off compared the performance of a P-processor system against P independent uniprocessor systems. For this situation, three different choices were made for the drum subsystem of the multiprocessor system. When the drum most closely resembled the P drums of the independent systems, the throughput of the multiprocessor system was essentially equivalent to the total throughput of the independent systems. In the two other choices, the relative throughput was better or worse depending on the capability of the drum subsystem chosen. In effect, then, the multiprocessor system had significantly higher performance only when its drum subsystem was superior.

In all of the relationships examined, the deterministic model provided a simple and clean way to ascertain the effects of various changes to the system.

7.2 Future Work

There are many areas for future work. Extensions and improvements to the deterministic model could be made. Specifically, an approximation for processor queue wait, Q_p, could cause the greatest improvement in model accuracy. Also, it would allow detailed examination of the operation of the system for the important case when it is approaching processor saturation, but not fully saturated. Furthermore, it would allow better analytic evaluation of the processor trade-offs. Another area for improvement is the inclusion of a term dealing with K, the squared coefficient of variation of the fault process. Although the model is quite accurate even without a value for K, this improvement would improve the accuracy for single processor systems and for abnormally bursty fault distributions.

Examination of the expression for a job's trip time could provide a reasonable measure of average turnaround. It is suspected that multiple processor systems might provide lower average turnaround than one processor systems.

The system model might also be adapted to describe other system configurations. For instance, systems involving more than one drum or even moving head disks might be modeled. The explicit description of drum writes of overwritten pages could be included. Also, the effects of non-shared memory could be investigated. This situation requires a job queue for each processor and jobs would probably return to the processor on which they had started execution.

In addition to improvements in the system model, the fault model needs to be developed. The results of the system model seem to indicate that an extremely detailed description of the faulting process is not necessary, since even using the mean of the fault distribution yields a reasonably good prediction of system performance. A simple model which relates the fault process to program behavior, memory size and page size, and job allocation strategy could, in combination with the system model developed here, allow a designer to examine many of the trade-offs inherent in computer system design in a quick and convenient manner. Also, the model might be compared against the operation of dynamic memory allocation schemes, with job preemption and restart, and revised if necessary.

APPENDIX

This appendix lists the results from the simulation runs. The input parameters are:

P   number of processors,

S   number of drum sectors,

J   number of jobs,

C   operating system overhead time per fault,

$1/\lambda$ interfault time, and

K   squared coefficient of variation of the

        fault process.

The outputs of the simulation are:

$1/\lambda_{sim}$ the minimum and maximum possible

        values of $1/\lambda$ from the simulation,

Q_p mean wait time in the processor queue,

Q_d mean wait time in a drum queue, and

T_{sim} throughput of the system expressed in

        mean number of usefully busy processors.

In addition, the throughput predicted by the deterministic model, T_{model}, is also included for reference.

P	S	J	C	$1/\lambda$	K	$1/\lambda_{sim}$		Q_p	Q_d	T_{sim}	T_{model}
16	32	16	0	8.000	1	8.022–	8.029	0.0	27.6	3.51	3.45
16	32	20	0	8.000	1	8.024–	8.029	0.0	31.3	3.98	3.94
16	32	32	0	8.000	1	8.025–	8.032	0.0	41.5	5.06	4.98
16	32	100	0	8.000	1	8.024–	8.039	0.5	103.8	6.99	6.82
16	32	16	0	32.000	1	31.633–	31.793	0.0	22.2	9.23	9.23
16	32	20	0	32.000	1	31.544–	31.763	0.0	23.9	11.19	11.13
16	32	32	0	32.000	1	31.489–	31.786	4.3	29.3	15.29	15.83
16	32	100	0	32.000	1	31.492–	31.789	132.8	31.1	16.00	16.00
16	32	16	0	128.000	1	123.684–	127.684	0.0	18.3	13.90	13.94
16	32	20	0	128.000	1	123.437–	127.725	14.3	17.6	15.91	16.00
16	32	32	0	128.000	1	123.080–	127.661	108.4	17.0	16.00	16.00
16	32	100	0	128.000	1	123.080–	127.661	588.3	17.0	16.00	16.00
16	8	16	0	8.000	1	8.023–	8.032	0.0	13.1	5.80	5.65
16	8	20	0	8.000	1	8.023–	8.036	0.0	16.5	6.28	6.13
16	8	32	0	8.000	1	8.022–	8.038	0.0	27.5	7.02	6.87
16	8	100	0	8.000	1	8.022–	8.039	0.6	91.4	7.87	7.66
16	8	16	0	32.000	1	31.505–	31.783	0.0	6.6	12.92	12.90
16	8	20	0	32.000	1	31.490–	31.787	1.6	7.5	15.19	15.67
16	8	32	0	32.000	1	31.492–	31.789	22.4	8.3	16.00	16.00
16	8	100	0	32.000	1	31.492–	31.789	153.7	8.3	16.00	16.00
16	8	16	0	128.000	1	123.186–	127.772	0.0	4.5	15.34	15.34
16	8	20	0	128.000	1	123.076–	127.657	26.4	4.6	16.00	16.00
16	8	32	0	128.000	1	123.080–	127.661	120.0	4.5	16.00	16.00
16	8	100	0	128.000	1	123.080–	127.661	595.5	4.5	16.00	16.00
1	32	16	0	0.500	1	0.509–	0.509	0.1	30.8	0.25	0.24
1	32	20	0	0.500	1	0.510–	0.510	0.1	34.6	0.28	0.27
1	32	100	0	0.500	1	0.507–	0.507	0.4	111.8	0.44	0.43
1	32	16	0	2.000	1	2.011–	2.011	5.2	28.5	0.87	0.95
1	32	20	0	2.000	1	2.011–	2.011	9.1	30.2	0.95	1.00
1	32	32	0	2.000	1	2.011–	2.011	29.9	31.5	1.00	1.00
1	32	100	0	2.000	1	2.011–	2.011	166.0	31.6	1.00	1.00
1	32	16	0	8.000	1	7.873–	7.873	98.7	18.0	1.00	1.00
1	32	20	0	8.000	1	7.873–	7.873	129.9	18.0	1.00	1.00
1	32	32	0	8.000	1	7.873–	7.873	223.1	18.0	1.00	1.00
1	32	16	0	0.125	1	0.127–	0.127	0.0	31.0	0.06	0.06
1	32	20	0	0.125	1	0.127–	0.127	0.0	34.9	0.07	0.07
1	32	32	0	0.125	1	0.127–	0.127	0.0	46.6	0.09	0.08
1	32	100	0	0.125	1	0.127–	0.127	0.1	112.2	0.11	0.11
1	8	16	0	0.500	1	0.504–	0.504	0.1	18.1	0.41	0.39
1	8	20	0	0.500	1	0.504–	0.504	0.1	21.9	0.43	0.41
1	8	32	0	0.500	1	0.504–	0.504	0.2	33.8	0.45	0.44
1	8	100	0	0.500	1	0.504–	0.504	0.7	98.4	0.50	0.48
1	8	16	0	2.000	1	2.011–	2.011	21.2	8.0	1.00	1.00
1	8	20	0	2.000	1	2.011–	2.011	29.2	8.0	1.00	1.00
1	8	16	0	8.000	1	7.873–	7.873	112.1	4.5	1.00	1.00
1	8	16	0	0.125	1	0.127–	0.127	0.0	18.6	0.10	0.10
4	32	4	0	8.000	1	8.024–	8.026	0.0	18.1	1.18	1.16
4	32	5	0	8.000	1	8.025–	8.026	0.0	19.0	1.43	1.41
4	32	8	0	8.000	1	8.023–	8.027	0.2	21.0	2.12	2.09

P	S	J	C	$1/\lambda$	K	$1/\lambda_{sim}$		Q_p	Q_d	T_{sim}	T_{model}
4	32	32	0	8.000	1	8.024-	8.028	23.1	32.1	3.99	4.00
4	32	4	0	32.000	1	31.828-	31.868	0.0	17.3	2.54	2.54
4	32	5	0	32.000	1	31.842-	31.863	1.3	17.1	3.11	3.15
4	32	8	0	32.000	1	31.799-	31.859	14.2	18.8	3.87	4.00
4	32	32	0	32.000	1	31.795-	31.856	202.1	18.1	4.00	4.00
4	32	4	0	128.000	1	126.487-	127.118	0.0	16.9	3.50	3.52
4	32	5	0	128.000	1	126.558-	127.504	18.8	17.0	3.88	4.00
4	32	8	0	128.000	1	126.496-	127.442	108.8	17.4	4.00	4.00
4	32	32	0	128.000	1	126.503-	127.450	842.5	17.1	4.00	4.00
4	8	8	0	8.000	1	8.024-	8.028	2.0	7.0	3.57	3.84
4	8	32	0	8.000	1	8.024-	8.027	46.9	8.1	4.00	4.00
4	8	8	0	32.000	1	31.795-	31.856	26.2	4.7	4.00	4.00
4	8	32	0	32.000	1	31.795-	31.856	215.4	4.4	4.00	4.00
4	8	8	0	128.000	1	126.503-	127.450	122.0	4.1	4.00	4.00
4	8	32	0	128.000	1	126.503-	127.450	854.6	4.1	4.00	4.00
1	32	8	0	2.000	1	2.011-	2.011	1.5	22.2	0.60	0.61
1	32	32	0	2.000	1	2.011-	2.011	29.9	31.5	1.00	1.00
1	32	4	0	8.000	1	7.873-	7.873	9.4	17.7	0.88	1.00
1	32	8	0	8.000	1	7.873-	7.873	35.8	18.5	1.00	1.00
1	32	4	0	32.000	1	31.902-	31.902	78.6	16.9	1.00	1.00
16	16	20	0	8.000	1	8.023-	8.031	0.0	21.5	5.26	5.12
16	16	32	0	8.000	1	8.023-	8.032	0.0	32.3	6.20	6.07
16	16	20	0	32.000	1	31.495-	31.772	0.3	13.8	13.56	13.63
16	16	32	0	32.000	1	31.491-	31.788	14.5	16.3	15.99	16.00
16	16	20	0	128.000	1	123.089-	127.681	21.8	9.3	15.99	16.00
16	16	32	0	128.000	1	123.580-	127.661	115.3	9.7	16.00	16.00
4	16	8	0	8.000	1	8.023-	8.027	0.7	11.9	2.97	2.95
4	16	32	0	8.000	1	8.024-	8.027	39.1	16.0	4.00	4.00
4	16	8	0	32.000	1	31.795-	31.855	22.0	9.1	3.99	4.00
4	16	32	0	32.000	1	31.795-	31.856	210.5	9.4	4.00	4.00
4	16	8	0	128.000	1	126.503-	127.450	117.8	8.3	4.00	4.00
4	16	32	0	128.000	1	126.503-	127.450	850.6	8.3	4.00	4.00
1	16	16	0	0.500	1	0.508-	0.508	0.1	22.3	0.34	0.33
1	16	20	0	0.500	1	0.508-	0.508	0.1	26.3	0.36	0.35
1	16	16	0	2.000	1	2.011-	2.011	13.8	15.6	0.99	1.00
1	16	16	0	8.000	1	7.873-	7.873	107.5	9.1	1.00	1.00
16	32	16	8	8.000	1	8.023-	8.030	0.0	25.1	3.05	3.00
16	32	20	8	8.000	1	8.024-	8.028	0.0	28.2	3.55	3.51
16	32	32	8	8.000	1	8.025-	8.032	0.0	37.9	4.67	4.64
16	32	100	8	8.000	1	8.022-	8.033	1.9	97.2	6.84	6.74
16	32	16	8	32.000	1	31.602-	31.782	0.0	21.2	8.20	8.19
16	32	20	8	32.000	1	31.598-	31.817	0.1	23.2	9.91	9.96
16	32	32	8	32.000	1	31.503-	31.761	13.1	25.9	12.74	12.80
16	32	100	8	32.000	1	31.509-	31.767	179.2	25.9	12.78	12.80
16	32	16	8	128.000	1	123.785-	127.788	0.0	17.8	13.22	13.23
16	32	20	8	128.000	1	123.502-	127.792	15.7	18.2	14.99	15.06
16	32	32	8	128.000	1	123.588-	127.882	115.9	18.1	15.06	15.06
16	32	100	8	128.000	1	123.588-	127.882	628.2	18.1	15.06	15.06
16	8	16	8	8.000	1	8.023-	8.031	0.0	9.4	4.87	4.76

P	S	J	C	$1/\lambda$	K	$1/\lambda_{sim}$		Q_p	Q_d	T_{sim}	T_{model}
16	8	20	8	8.000	1	8.024–	8.031	0.0	11.9	5.55	5.43
16	8	32	8	8.000	1	8.022–	8.034	0.2	21.2	6.68	6.55
16	8	100	8	8.000	1	8.024–	8.036	8.5	77.1	7.77	7.64
16	8	16	8	32.000	1	31.560–	31.798	0.0	6.0	10.87	10.88
16	8	20	8	32.000	1	31.502–	31.760	3.3	6.5	12.58	12.80
16	8	32	8	32.000	1	31.509–	31.767	32.0	6.7	12.78	12.80
16	8	100	8	32.000	1	31.509–	31.767	196.4	6.7	12.78	12.80
16	8	16	8	128.000	1	123.877–	127.883	0.0	4.4	14.49	14.47
16	8	20	8	128.000	1	123.588–	127.882	28.6	4.5	15.06	15.06
16	8	32	8	128.000	1	123.588–	127.882	128.5	4.5	15.06	15.06
16	8	100	8	128.000	1	123.588–	127.882	636.1	4.5	15.06	15.06
16	32	16	0	8.000	16	7.924–	7.924	0.0	27.6	3.47	3.45
16	32	20	0	8.000	16	7.906–	7.906	0.0	31.4	3.92	3.94
16	32	32	0	8.000	16	7.860–	7.882	0.0	42.0	4.94	4.98
16	32	100	0	8.000	16	7.830–	7.862	0.0	104.0	6.87	6.82
16	32	16	0	32.000	16	29.584–	30.372	0.0	22.7	8.99	9.23
16	32	20	0	32.000	16	29.251–	30.313	0.1	24.7	10.81	11.13
16	32	32	0	32.000	16	29.320–	30.385	5.7	32.3	13.97	15.83
16	32	100	0	32.000	16	29.282–	30.345	108.0	47.7	15.93	16.00
16	32	16	0	127.996	16	117.022–	121.269	0.0	18.0	13.84	13.94
16	32	20	0	127.996	16	119.962–	122.871	14.7	19.8	15.51	16.00
16	32	32	0	127.996	16	100.206–	110.631	87.4	23.3	15.85	16.00
16	32	100	0	127.996	16	100.296–	110.731	531.6	27.4	16.00	16.00
16	8	16	0	8.000	16	7.835–	7.867	0.0	13.2	5.69	5.65
16	8	20	0	8.000	16	7.830–	7.863	0.0	16.7	6.15	6.13
16	8	32	0	8.000	16	7.825–	7.858	0.0	27.6	6.88	6.87
16	8	100	0	8.000	16	7.813–	7.856	0.0	91.9	7.71	7.66
16	8	16	0	32.000	16	30.084–	30.813	0.0	6.9	12.72	12.90
16	8	20	0	32.000	16	30.002–	30.737	2.0	8.8	14.45	15.67
16	8	32	0	32.000	16	29.973–	30.700	16.2	14.9	15.59	16.00
16	8	100	0	32.000	16	29.944–	30.670	138.6	19.9	15.97	16.00
16	8	16	0	127.996	16	117.299–	121.557	0.0	4.9	15.26	15.34
16	8	20	0	127.996	16	100.217–	110.643	21.3	6.6	15.84	16.00
16	8	32	0	127.996	16	100.229–	110.657	101.7	7.8	15.93	16.00
16	8	100	0	127.996	16	100.305–	110.741	547.2	10.9	16.00	16.00
1	32	16	0	0.500	16	0.486–	0.486	0.9	30.6	0.24	0.24
1	32	20	0	0.500	16	0.486–	0.486	1.1	34.5	0.26	0.27
1	32	32	0	0.500	16	0.486–	0.486	1.5	45.9	0.32	0.33
1	32	100	0	0.500	16	0.486–	0.486	2.6	110.2	0.42	0.43
1	32	16	0	2.000	16	1.921–	1.921	14.7	27.5	0.68	0.95
1	32	20	0	2.000	16	1.921–	1.921	18.7	30.1	0.74	1.00
1	32	32	0	2.000	16	1.961–	1.961	32.4	35.8	0.88	1.00
1	32	100	0	2.000	16	1.961–	1.961	146.5	44.7	1.00	1.00
1	32	16	0	8.000	16	7.681–	7.681	96.8	24.4	0.95	1.00
1	32	20	0	8.000	16	7.681–	7.681	124.3	25.4	0.97	1.00
1	32	32	0	8.000	16	7.681–	7.681	210.7	26.9	0.99	1.00
1	8	16	0	0.500	16	0.486–	0.486	1.7	17.4	0.38	0.39
1	8	20	0	0.500	16	0.486–	0.486	1.9	21.1	0.40	0.41
1	8	32	0	0.500	16	0.486–	0.486	2.3	32.4	0.43	0.44

MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

right

88
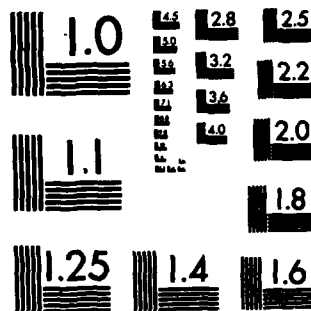
P	S	J	C	$1/\lambda$	K	$1/\lambda_{sim}$		Q_p	Q_d	T_{sim}	T_{model}
1	8	100	0	0.500	16	0.486–	0.486	3.2	97.1	0.47	0.48
1	8	16	0	2.000	16	1.921–	1.921	20.3	13.3	0.84	1.00
1	8	20	0	2.000	16	1.921–	1.921	25.7	15.1	0.88	1.00
1	8	32	0	2.000	16	1.961–	1.961	44.4	18.3	0.95	1.00
1	8	100	0	2.000	16	1.961–	1.961	167.9	22.5	1.00	1.00
1	8	16	0	8.000	16	7.681–	7.681	105.7	10.9	0.98	1.00
1	8	20	0	8.000	16	7.681–	7.681	134.9	11.6	0.99	1.00
4	32	4	0	8.000	16	8.007–	8.007	0.0	18.3	1.17	1.16
4	32	5	0	8.000	16	7.995–	7.995	0.1	19.0	1.43	1.41
4	32	8	0	8.000	16	7.963–	7.963	0.8	21.5	2.04	2.09
4	32	32	0	8.000	16	7.846–	7.878	24.3	35.9	3.64	4.00
4	32	4	0	32.000	16	31.256–	31.408	0.0	17.0	2.54	2.54
4	32	5	0	32.000	16	31.197–	31.349	2.7	17.7	2.97	3.15
4	32	8	0	32.000	16	31.188–	31.339	18.1	19.9	3.56	4.00
4	32	32	0	32.000	16	31.165–	31.317	191.0	25.8	4.00	4.00
4	32	4	0	127.996	16	116.295–	116.951	0.0	16.4	3.48	3.52
4	32	5	0	127.996	16	118.494–	119.426	21.8	16.9	3.75	4.00
4	32	8	0	127.996	16	119.187–	120.076	103.2	18.6	3.95	4.00
4	32	32	0	127.996	16	121.262–	122.154	822.2	20.1	4.00	4.00
4	8	8	0	8.000	16	7.874–	7.885	3.2	8.0	3.14	3.84
4	8	32	0	8.000	16	7.839–	7.872	37.8	18.0	3.89	4.00
4	8	8	0	32.000	16	31.173–	31.324	25.1	6.8	3.90	4.00
4	8	32	0	32.000	16	31.168–	31.319	206.9	9.6	4.00	4.00
4	8	8	0	127.996	16	121.264–	122.155	115.6	5.3	4.00	4.00
4	8	32	0	127.996	16	120.410–	121.300	833.2	5.3	4.00	4.00
1	32	8	0	2.000	16	1.957–	1.957	6.8	22.0	0.49	0.61
1	32	32	0	2.000	16	1.957–	1.957	32.2	36.0	0.88	1.00
1	32	4	0	8.000	16	7.681–	7.681	18.6	18.8	0.67	1.00
1	32	8	0	8.000	16	7.843–	7.843	44.6	21.3	0.84	1.00
1	32	4	0	32.000	16	29.844–	29.844	83.0	18.4	0.90	1.00
16	16	20	0	8.000	16	7.430–	7.610	0.0	21.5	5.05	5.12
16	16	32	0	8.000	16	7.419–	7.617	0.0	31.8	6.00	6.07
16	16	20	0	32.000	16	30.030–	30.758	0.7	14.3	13.16	13.63
16	16	32	0	32.000	16	29.994–	30.722	11.9	20.7	15.24	16.00
16	16	20	0	127.996	16	117.218–	121.473	20.1	11.1	15.80	16.00
16	16	32	0	127.996	16	117.127–	121.378	107.5	12.1	15.96	16.00
16	32	32	8	8.000	16	7.435–	7.615	0.0	37.3	4.50	4.64
16	32	100	8	8.000	16	7.427–	7.606	3.0	94.7	6.55	6.74
16	32	32	8	32.000	16	29.298–	30.290	11.9	28.5	12.16	12.80
16	32	100	8	32.000	16	29.323–	30.316	164.9	32.2	12.66	12.80
16	32	32	8	127.996	16	117.267–	121.523	107.9	20.0	14.99	15.06
16	32	100	8	127.996	16	117.267–	121.523	643.1	20.0	15.01	15.06
16	8	32	8	8.000	16	7.431–	7.611	0.3	21.1	6.40	6.55
16	8	100	8	8.000	16	7.426–	7.606	15.1	71.9	7.27	7.64
16	8	32	8	32.000	16	29.323–	30.316	29.6	7.8	12.63	12.80
16	8	100	8	32.000	16	29.323–	30.316	188.4	8.2	12.66	12.80
16	8	32	8	127.996	16	117.267–	121.523	122.3	5.1	15.01	15.06
16	8	100	8	127.996	16	117.267–	121.523	657.8	5.0	15.01	15.06

REFERENCES

[ADAM79] Adams, C., E. Gelenbe, and J. Vicard. "An Experimentally Validated Model of the Paging Drum," Acta Informatica, 1979, pp. 103-117.

[BIRT73] Birtwistle, Graham M., Ole-Johan Dahl, Bjorn Myhrhaug, and Kristen Nygaard. Simula Begin, Van Nostrand Reinhold, 1973.

[BLAK79] Blake, Russ. "Tailor: A Simple Model That Works," 1979 Conference on Simulation, Measurement and Modeling of Computer Systems, August, 1979, pp. 1-12.

[BRIG77a] Briggs, Faye Alaye. "Memory Organizations and Their Effectiveness for Multiprocessing Computers," Coordinated Science Laboratory Report R-768, University of Illinois, 1977.

[BRIG77b] Briggs, F.A., and E.S. Davidson. "Organization of Semiconductor Memories for Parallel-Pipelined Processors," IEEE Transactions on Computers, February, 1977, pp. 162-169.

[BROW73] Brown, J.C., K.M. Chandy, J. Hogarth, and C.C.-A. Lee. "The Effect on Throughput of Multiprocessing in a Multiprogramming Environment," IEEE Transactions on Computers, August, 1973, pp. 728-735.

[BUDZ77] Budzinski, Robert Lucius. "Dynamic Memory Allocation for a Virtual Memory Computer," Coordinated Science Laboratory Report R-754, University of Illinois, 1977.

[CDC75] Control Data Corporation. Simula Version 1 Reference Manual, Control Data, 1975.

[CHAN75] Chandy, K.M., U. Herzog, and L. Woo. "Approximate Analysis of General Queueing Networks," IBM Journal of Research Development, January, 1975, pp. 43-49.

[CHAN77] Chandy, K. Mani, John H. Howard, Jr., and Donald F. Towsley. "Product Form and Local Balance in Queueing Networks," Journal of the ACM, April, 1977, pp. 250-263.

[CHU72]   Chu, W.W., and H. Opderbeck. "The Page Fault Frequency
           Replacement Algorithm," Proceedings FJCC 1972, 1972, pp.
           597-609.

[COFF69]   Coffman, E.G., Jr. "Analysis of a Drum Input/Output Queue
           Under Scheduled Operation in a Paged Computer System," Journal
           of the ACM, January, 1969, pp. 73-90.

[COFF73]   Coffman, E.G., Jr., and P.J. Denning. Operating Systems
           Theory, Prentice-Hall, 1973.

[DENN68]   Denning, P.J. "The Working Set Model for Program Behavior,"
           Communications of the ACM, May, 1968, pp.323-333.

[FERR76]   Ferrari, Domenico. "The Improvement of Program Behavior,"
           Computer, November, 1976, pp. 39-47.

[FULL75]   Fuller, Samuel H., and Forest Baskett. "An Analysis of Drum
           Storage Units," Journal of the ACM, January, 1975, pp.
           83-105.

[GELE75]   Gelenbe, Erol. "On Approximate Computer System Models,"
           Journal of the ACM, April, 1975, pp. 261-269.

[JACK63]   Jackson, J.R. "Job-shop Like Queueing Systems," Management
           Science, 1963, pp. 131-142.

[KLEI75]   Kleinrock, Leonard. Queueing Systems, Volume I: Theory, John
           Wiley and Sons, 1975.

[KOBA74]   Kobayashi, Hisashi. "Application of the Diffusion
           Approximation to Queueing Networks I: Equilibrium Queue
           Distributions," Journal of the ACM, April, 1974, pp.316-328.

[KUCK70]   Kuck, D.J., and D.H. Lawrie. "The Use and Performance of
           Memory Hierarchies: A Survey," Software Engineering, Vol. 1,
           J.T. Tou, ed., Academic Press, 1970, pp. 45-77.

[LAZO77]   Lazowska, Edward D. "The Use of Percentiles in Characterizing
           Service Time Distributions," Proceedings 1977 International
           Symposium on Computer Performance Modeling, Measurement, and
           Evaluation, August, 1977.

[SAUE79]   Sauer, Charles H., and K. Mani Chandy. "The Impact of
           Distributions and Disciplines on Multiple Processor Systems,"
           Communications of the ACM, January, 1979, pp. 25-34.

VITA

Alan Dale Gant was born in Wichita Falls, Texas on January 17, 1951. He received the B.S. degree in Electrical Engineering with highest honors from the University of Texas in 1974, and the M.S. degree in Electrical Engineering from the University of Illinois in 1977. He held engineering positions with IBM in Austin, Texas and Mostek in Carrollton, Texas. While at the University of Illinois, he held a University Fellowship in 1974 and was a research assistant at the Coordinated Science Laboratory from 1975 to 1980.

8